

CF User's Guide

Generated by Doxygen 1.8.17

1 CFDP (CF) Documentation	1
1.1 CFS CFDP Introduction	2
1.2 CFS CFDP Overview	3
1.3 CFS CFDP Design	4
1.4 CFS CFDP Operation	4
1.5 CFS CFDP Deployment Guide	10
1.6 CFS CFDP Configuration	11
1.7 CFS CFDP Table Definitions	11
1.8 CFS CFDP Commands	11
1.8.1 CFS CFDP Command Descriptions	12
1.9 CFS CFDP Telemetry	23
1.9.1 CFS CFDP Telemetry Descriptions	23
1.10 CFS CFDP Events	24
1.11 CFS CFDP Constraints	24
1.12 CFS CFDP Frequently Asked Questions	25
1.13 CFS CFDP Lessons Learned	25
2 Core Flight Executive Documentation	26
2.1 Background	27
2.1.1 Core Flight Executive (cFE) Goals	28
2.2 Applicable Documents	28
2.3 Version Numbers	28
2.3.1 Version Number Semantics	28
2.3.2 How and Where Defined	29
2.3.3 Identifying Development Builds	29
2.3.4 Templates for the short and long version string	29
2.4 Dependencies	30
2.5 Acronyms	30
2.6 cFE Executive Services Overview	31
2.6.1 Terminology	32
2.6.2 Software Reset	34
2.6.3 Reset Types and Subtypes	35
2.6.4 Exception and Reset (ER) Log	35
2.6.5 Application and Child Task Management	35
2.6.6 Starting an Application	36
2.6.7 Stopping an Application	36
2.6.8 Restarting an Application	36
2.6.9 Reloading an Application	37
2.6.10 Listing Current Applications	37

2.6.11 Listing Current Tasks	38
2.6.12 Loading Common Libraries	38
2.6.13 Basic File System	38
2.6.14 Performance Data Collection	38
2.6.15 Critical Data Store	39
2.6.16 Memory Pool	40
2.6.17 System Log	42
2.6.18 Version Identification	42
2.6.19 Frequently Asked Questions about Executive Services	42
2.7 cFE Executive Services Commands	43
2.8 cFE Executive Services Telemetry	44
2.9 cFE Executive Services Configuration Parameters	45
2.10 cFE Event Services Overview	49
2.10.1 Event Message Format	50
2.10.2 Local Event Log	51
2.10.3 Event Message Control	51
2.10.4 Event Message Filtering	53
2.10.5 EVS Registry	54
2.10.6 EVS Counters	54
2.10.7 Resetting EVS Counters	55
2.10.8 Effects of a Processor Reset on EVS	55
2.10.9 EVS squelching of misbehaving apps	56
2.10.10 Frequently Asked Questions about Event Services	56
2.11 cFE Event Services Commands	57
2.12 cFE Event Services Telemetry	59
2.13 cFE Event Services Configuration Parameters	59
2.14 cFE Software Bus Overview	60
2.14.1 Software Bus Terminology	61
2.14.2 Autonomous Actions	62
2.14.3 Operation of the SB Software	63
2.14.4 Frequently Asked Questions about Software Bus	66
2.15 cFE Software Bus Commands	67
2.16 cFE Software Bus Telemetry	68
2.17 cFE Software Bus Configuration Parameters	69
2.18 cFE Table Services Overview	70
2.18.1 Managing Tables	70
2.18.2 cFE Table Types and Table Options	71
2.18.3 Table Registry	73
2.18.4 Table Services Telemetry	74

2.18.5 Effects of Processor Reset on Tables	74
2.18.6 Frequently Asked Questions about Table Services	74
2.19 cFE Table Services Commands	76
2.20 cFE Table Services Telemetry	76
2.21 cFE Table Services Configuration Parameters	77
2.22 cFE Time Services Overview	78
2.22.1 Time Components	80
2.22.2 Time Structure	80
2.22.3 Time Formats	81
2.22.4 Time Configuration	81
2.22.5 Time Format Selection	86
2.22.6 Enabling Fake Tone Signal	86
2.22.7 Selecting Tone and Data Ordering	86
2.22.8 Specifying Tone and Data Window	87
2.22.9 Specifying Time Server/Client	87
2.22.10 Specifying Time Tone Byte Order	87
2.22.11 Virtual MET	88
2.22.12 Specifying Time Source	88
2.22.13 Specifying Time Signal	89
2.22.14 Time Services Paradigm(s)	89
2.22.15 Flywheeling	90
2.22.16 Time State	90
2.22.17 Initialization	90
2.22.18 Power-On Reset	91
2.22.19 Processor Reset	91
2.22.20 Initialization	92
2.22.21 Power-On Reset	92
2.22.22 Processor Reset	93
2.22.23 Normal Operation	93
2.22.24 Client	95
2.22.25 Server	95
2.22.26 Setting Time	96
2.22.27 Adjusting Time	97
2.22.28 Setting MET	97
2.22.29 Frequently Asked Questions about Time Services	97
2.23 cFE Time Services Commands	97
2.24 cFE Time Services Telemetry	98
2.25 cFE Time Services Configuration Parameters	99
2.26 cFE Event Message Cross Reference	100

2.27 cFE Command Mnemonic Cross Reference	100
2.28 cFE Telemetry Mnemonic Cross Reference	104
3 Glossary of Terms	115
4 cFE Application Programmer's Interface (API) Reference	116
4.1 Executive Services API	116
4.2 Events Services API	118
4.3 File Services API	119
4.4 Message API	119
4.5 Resource ID API	120
4.6 Software Bus Services API	120
4.7 Table Services API	121
4.8 Time Services API	122
5 Osal API Documentation	123
5.1 OSAL Introduction	124
5.2 File System Overview	125
5.3 File Descriptors In Osal	125
5.4 Timer Overview	126
6 cFE Mission Configuration Parameters	126
7 Module Index	127
7.1 Modules	127
8 Data Structure Index	130
8.1 Data Structures	130
9 File Index	146
9.1 File List	146
10 Module Documentation	152
10.1 CFS CFDP Event IDs	152
10.1.1 Detailed Description	158
10.1.2 Macro Definition Documentation	158
10.2 CFS CFDP Mission Configuration	192
10.2.1 Detailed Description	192
10.2.2 Macro Definition Documentation	192
10.3 CFS CFDP Version	194
10.3.1 Detailed Description	194
10.3.2 Macro Definition Documentation	194

10.4 CFS CFDP Command Codes	195
10.4.1 Detailed Description	195
10.4.2 Enumeration Type Documentation	195
10.5 CFS CFDP Platform Configuration	216
10.5.1 Detailed Description	217
10.5.2 Macro Definition Documentation	217
10.6 CFS CFDP Telemetry	222
10.6.1 Detailed Description	223
10.6.2 Typedef Documentation	223
10.7 CFS CFDP Command Structures	224
10.7.1 Detailed Description	226
10.7.2 Typedef Documentation	226
10.7.3 Enumeration Type Documentation	229
10.8 CFS CFDP Command Message IDs	231
10.8.1 Detailed Description	231
10.8.2 Macro Definition Documentation	231
10.9 CFS CFDP Telemetry Message IDs	232
10.9.1 Detailed Description	232
10.9.2 Macro Definition Documentation	232
10.10 CFS CFDP Data Interface Message IDs	233
10.10.1 Detailed Description	233
10.10.2 Macro Definition Documentation	233
10.11 cFE Return Code Defines	234
10.11.1 Detailed Description	239
10.11.2 Macro Definition Documentation	239
10.12 cFE Resource ID APIs	257
10.12.1 Detailed Description	257
10.12.2 Function Documentation	257
10.13 cFE Entry/Exit APIs	260
10.13.1 Detailed Description	260
10.13.2 Function Documentation	260
10.14 cFE Application Control APIs	262
10.14.1 Detailed Description	262
10.14.2 Function Documentation	262
10.15 cFE Application Behavior APIs	265
10.15.1 Detailed Description	265
10.15.2 Function Documentation	265
10.16 cFE Information APIs	269
10.16.1 Detailed Description	269

10.16.2 Function Documentation	269
10.17 cFE Child Task APIs	278
10.17.1 Detailed Description	278
10.17.2 Function Documentation	278
10.18 cFE Miscellaneous APIs	282
10.18.1 Detailed Description	282
10.18.2 Function Documentation	282
10.19 cFE Critical Data Store APIs	285
10.19.1 Detailed Description	285
10.19.2 Function Documentation	285
10.20 cFE Memory Manager APIs	290
10.20.1 Detailed Description	290
10.20.2 Function Documentation	290
10.21 cFE Performance Monitor APIs	297
10.21.1 Detailed Description	297
10.21.2 Macro Definition Documentation	297
10.21.3 Function Documentation	298
10.22 cFE Generic Counter APIs	299
10.22.1 Detailed Description	299
10.22.2 Function Documentation	299
10.23 cFE Registration APIs	305
10.23.1 Detailed Description	305
10.23.2 Function Documentation	305
10.24 cFE Send Event APIs	307
10.24.1 Detailed Description	307
10.24.2 Function Documentation	307
10.25 cFE Reset Event Filter APIs	312
10.25.1 Detailed Description	312
10.25.2 Function Documentation	312
10.26 cFE File Header Management APIs	314
10.26.1 Detailed Description	314
10.26.2 Function Documentation	314
10.27 cFE File Utility APIs	318
10.27.1 Detailed Description	318
10.27.2 Function Documentation	318
10.28 cFE Generic Message APIs	323
10.28.1 Detailed Description	323
10.28.2 Function Documentation	323
10.29 cFE Message Primary Header APIs	324

10.29.1 Detailed Description	324
10.29.2 Function Documentation	324
10.30 cFE Message Extended Header APIs	333
10.30.1 Detailed Description	333
10.30.2 Function Documentation	333
10.31 cFE Message Secondary Header APIs	339
10.31.1 Detailed Description	339
10.31.2 Function Documentation	339
10.32 cFE Message Id APIs	344
10.32.1 Detailed Description	344
10.32.2 Function Documentation	344
10.33 cFE Message Integrity APIs	346
10.33.1 Detailed Description	346
10.33.2 Function Documentation	346
10.34 cFE Pipe Management APIs	348
10.34.1 Detailed Description	348
10.34.2 Function Documentation	348
10.35 cFE Message Subscription Control APIs	353
10.35.1 Detailed Description	353
10.35.2 Function Documentation	353
10.36 cFE Send/Receive Message APIs	358
10.36.1 Detailed Description	358
10.36.2 Function Documentation	358
10.37 cFE Zero Copy APIs	361
10.37.1 Detailed Description	361
10.37.2 Function Documentation	361
10.38 cFE Message Characteristics APIs	364
10.38.1 Detailed Description	364
10.38.2 Function Documentation	364
10.39 cFE Message ID APIs	369
10.39.1 Detailed Description	369
10.39.2 Function Documentation	369
10.40 cFE SB Pipe options	374
10.40.1 Detailed Description	374
10.40.2 Macro Definition Documentation	374
10.41 cFE Registration APIs	375
10.41.1 Detailed Description	375
10.41.2 Function Documentation	375
10.42 cFE Manage Table Content APIs	380

10.42.1 Detailed Description	380
10.42.2 Function Documentation	380
10.43 cFE Access Table Content APIs	386
10.43.1 Detailed Description	386
10.43.2 Function Documentation	386
10.44 cFE Get Table Information APIs	391
10.44.1 Detailed Description	391
10.44.2 Function Documentation	391
10.45 cFE Table Type Defines	394
10.45.1 Detailed Description	394
10.45.2 Macro Definition Documentation	394
10.46 cFE Get Current Time APIs	396
10.46.1 Detailed Description	396
10.46.2 Function Documentation	396
10.47 cFE Get Time Information APIs	399
10.47.1 Detailed Description	399
10.47.2 Function Documentation	399
10.48 cFE Time Arithmetic APIs	402
10.48.1 Detailed Description	402
10.48.2 Function Documentation	402
10.49 cFE Time Conversion APIs	405
10.49.1 Detailed Description	405
10.49.2 Function Documentation	405
10.50 cFE External Time Source APIs	407
10.50.1 Detailed Description	407
10.50.2 Function Documentation	407
10.51 cFE Miscellaneous Time APIs	412
10.51.1 Detailed Description	412
10.51.2 Function Documentation	412
10.52 cFE Resource ID base values	415
10.52.1 Detailed Description	415
10.52.2 Enumeration Type Documentation	415
10.53 cFE Clock State Flag Defines	417
10.53.1 Detailed Description	417
10.53.2 Macro Definition Documentation	417
10.54 OSAL Semaphore State Defines	419
10.54.1 Detailed Description	419
10.54.2 Macro Definition Documentation	419
10.55 OSAL Binary Semaphore APIs	420

10.55.1 Detailed Description	420
10.55.2 Function Documentation	420
10.56 OSAL BSP low level access APIs	425
10.56.1 Detailed Description	425
10.56.2 Function Documentation	425
10.57 OSAL Real Time Clock APIs	426
10.57.1 Detailed Description	427
10.57.2 Function Documentation	427
10.58 OSAL Core Operation APIs	440
10.58.1 Detailed Description	440
10.58.2 Function Documentation	440
10.59 OSAL Condition Variable APIs	444
10.59.1 Detailed Description	444
10.59.2 Function Documentation	444
10.60 OSAL Counting Semaphore APIs	450
10.60.1 Detailed Description	450
10.60.2 Function Documentation	450
10.61 OSAL Directory APIs	455
10.61.1 Detailed Description	455
10.61.2 Function Documentation	455
10.62 OSAL Return Code Defines	459
10.62.1 Detailed Description	461
10.62.2 Macro Definition Documentation	461
10.63 OSAL Error Info APIs	466
10.63.1 Detailed Description	466
10.63.2 Function Documentation	466
10.64 OSAL File Access Option Defines	468
10.64.1 Detailed Description	468
10.64.2 Macro Definition Documentation	468
10.65 OSAL Reference Point For Seek Offset Defines	469
10.65.1 Detailed Description	469
10.65.2 Macro Definition Documentation	469
10.66 OSAL Standard File APIs	470
10.66.1 Detailed Description	470
10.66.2 Function Documentation	470
10.67 OSAL File System Level APIs	483
10.67.1 Detailed Description	483
10.67.2 Function Documentation	483
10.68 OSAL Heap APIs	491

10.68.1 Detailed Description	491
10.68.2 Function Documentation	491
10.69 OSAL Object Type Defines	492
10.69.1 Detailed Description	492
10.69.2 Macro Definition Documentation	492
10.70 OSAL Object ID Utility APIs	495
10.70.1 Detailed Description	495
10.70.2 Function Documentation	495
10.71 OSAL Dynamic Loader and Symbol APIs	500
10.71.1 Detailed Description	500
10.71.2 Function Documentation	500
10.72 OSAL Mutex APIs	504
10.72.1 Detailed Description	504
10.72.2 Function Documentation	504
10.73 OSAL Network ID APIs	508
10.73.1 Detailed Description	508
10.73.2 Function Documentation	508
10.74 OSAL Printf APIs	510
10.74.1 Detailed Description	510
10.74.2 Function Documentation	510
10.75 OSAL Message Queue APIs	511
10.75.1 Detailed Description	511
10.75.2 Function Documentation	511
10.76 OSAL Select APIs	515
10.76.1 Detailed Description	515
10.76.2 Function Documentation	515
10.77 OSAL Shell APIs	521
10.77.1 Detailed Description	521
10.77.2 Function Documentation	521
10.78 OSAL Socket Address APIs	522
10.78.1 Detailed Description	522
10.78.2 Function Documentation	522
10.79 OSAL Socket Management APIs	526
10.79.1 Detailed Description	526
10.79.2 Function Documentation	526
10.80 OSAL Task APIs	536
10.80.1 Detailed Description	536
10.80.2 Function Documentation	536
10.81 OSAL Time Base APIs	542

10.81.1 Detailed Description	542
10.81.2 Function Documentation	542
10.82 OSAL Timer APIs	547
10.82.1 Detailed Description	547
10.82.2 Function Documentation	547
11 Data Structure Documentation	553
11.1 CCSDS_ExtendedHeader Struct Reference	553
11.1.1 Detailed Description	553
11.1.2 Field Documentation	553
11.2 CCSDS_PrimaryHeader Struct Reference	553
11.2.1 Detailed Description	553
11.2.2 Field Documentation	554
11.3 CF_AbandonCmd Struct Reference	554
11.3.1 Detailed Description	554
11.3.2 Field Documentation	554
11.4 CF_AppData_t Struct Reference	555
11.4.1 Detailed Description	555
11.4.2 Field Documentation	555
11.5 CF_CancelCmd Struct Reference	556
11.5.1 Detailed Description	556
11.5.2 Field Documentation	556
11.6 CF_CFDP_CycleTx_args Struct Reference	557
11.6.1 Detailed Description	557
11.6.2 Field Documentation	557
11.7 CF_CFDP_FileDirectiveDispatchTable_t Struct Reference	557
11.7.1 Detailed Description	558
11.7.2 Field Documentation	558
11.8 CF_CFDP_Jv Struct Reference	558
11.8.1 Detailed Description	558
11.8.2 Field Documentation	558
11.9 CF_CFDP_PduAck Struct Reference	558
11.9.1 Detailed Description	559
11.9.2 Field Documentation	559
11.10 CF_CFDP_PduEof Struct Reference	559
11.10.1 Detailed Description	559
11.10.2 Field Documentation	559
11.11 CF_CFDP_PduFileDialogContent Struct Reference	560
11.11.1 Detailed Description	560

11.11.2 Field Documentation	560
11.12 CF_CFDP_PduFileDataHeader Struct Reference	560
11.12.1 Detailed Description	560
11.12.2 Field Documentation	561
11.13 CF_CFDP_PduFileDirectiveHeader Struct Reference	561
11.13.1 Detailed Description	561
11.13.2 Field Documentation	561
11.14 CF_CFDP_PduFin Struct Reference	561
11.14.1 Detailed Description	561
11.14.2 Field Documentation	562
11.15 CF_CFDP_PduHeader Struct Reference	562
11.15.1 Detailed Description	562
11.15.2 Field Documentation	562
11.16 CF_CFDP_PduMd Struct Reference	563
11.16.1 Detailed Description	563
11.16.2 Field Documentation	563
11.17 CF_CFDP_PduNak Struct Reference	563
11.17.1 Detailed Description	563
11.17.2 Field Documentation	564
11.18 CF_CFDP_R_SubstateDispatchTable_t Struct Reference	564
11.18.1 Detailed Description	564
11.18.2 Field Documentation	564
11.19 CF_CFDP_S_SubstateRecvDispatchTable_t Struct Reference	564
11.19.1 Detailed Description	565
11.19.2 Field Documentation	565
11.20 CF_CFDP_S_SubstateSendDispatchTable_t Struct Reference	565
11.20.1 Detailed Description	565
11.20.2 Field Documentation	565
11.21 CF_CFDP_SegmentRequest Struct Reference	565
11.21.1 Detailed Description	566
11.21.2 Field Documentation	566
11.22 CF_CFDP_Tick_args Struct Reference	566
11.22.1 Detailed Description	566
11.22.2 Field Documentation	566
11.23 CF_CFDP_tlv Struct Reference	567
11.23.1 Detailed Description	567
11.23.2 Field Documentation	567
11.24 CF_CFDP_TxnRecvDispatchTable_t Struct Reference	568
11.24.1 Detailed Description	568

11.24.2 Field Documentation	568
11.25 CF_CFDP_TxnSendDispatchTable_t Struct Reference	568
11.25.1 Detailed Description	568
11.25.2 Field Documentation	568
11.26 CF_CFDP_uint16_t Struct Reference	569
11.26.1 Detailed Description	569
11.26.2 Field Documentation	569
11.27 CF_CFDP_uint32_t Struct Reference	569
11.27.1 Detailed Description	569
11.27.2 Field Documentation	569
11.28 CF_CFDP_uint64_t Struct Reference	570
11.28.1 Detailed Description	570
11.28.2 Field Documentation	570
11.29 CF_CFDP_uint8_t Struct Reference	570
11.29.1 Detailed Description	570
11.29.2 Field Documentation	570
11.30 CF_ChanAction_BoolArg Struct Reference	570
11.30.1 Detailed Description	571
11.30.2 Field Documentation	571
11.31 CF_ChanAction_BoolMsgArg Struct Reference	571
11.31.1 Detailed Description	571
11.31.2 Field Documentation	571
11.32 CF_ChanAction_MsgArg Struct Reference	571
11.32.1 Detailed Description	572
11.32.2 Field Documentation	572
11.33 CF_ChanAction_SuspResArg Struct Reference	572
11.33.1 Detailed Description	572
11.33.2 Field Documentation	572
11.34 CF_Channel Struct Reference	573
11.34.1 Detailed Description	573
11.34.2 Field Documentation	573
11.35 CF_ChannelConfig Struct Reference	574
11.35.1 Detailed Description	575
11.35.2 Field Documentation	575
11.36 CF_Chunk Struct Reference	577
11.36.1 Detailed Description	577
11.36.2 Field Documentation	577
11.37 CF_ChunkList Struct Reference	577
11.37.1 Detailed Description	578

11.37.2 Field Documentation	578
11.38 CF_ChunkWrapper Struct Reference	578
11.38.1 Detailed Description	578
11.38.2 Field Documentation	579
11.39 CF_CListNode Struct Reference	579
11.39.1 Detailed Description	579
11.39.2 Field Documentation	579
11.40 CF_Codec_BitField Struct Reference	579
11.40.1 Detailed Description	580
11.40.2 Field Documentation	580
11.41 CF_CodecState Struct Reference	580
11.41.1 Detailed Description	580
11.41.2 Field Documentation	580
11.42 CF_ConfigTable Struct Reference	581
11.42.1 Detailed Description	581
11.42.2 Field Documentation	581
11.43 CF_Crc Struct Reference	582
11.43.1 Detailed Description	583
11.43.2 Field Documentation	583
11.44 CF_DecoderState Struct Reference	583
11.44.1 Detailed Description	583
11.44.2 Field Documentation	583
11.45 CF_DisableDequeueCmd Struct Reference	584
11.45.1 Detailed Description	584
11.45.2 Field Documentation	584
11.46 CF_DisableDirPollingCmd Struct Reference	584
11.46.1 Detailed Description	585
11.46.2 Field Documentation	585
11.47 CF_DisableEngineCmd Struct Reference	585
11.47.1 Detailed Description	585
11.47.2 Field Documentation	585
11.48 CF_EnableDequeueCmd Struct Reference	585
11.48.1 Detailed Description	586
11.48.2 Field Documentation	586
11.49 CF_EnableDirPollingCmd Struct Reference	586
11.49.1 Detailed Description	586
11.49.2 Field Documentation	586
11.50 CF_EnableEngineCmd Struct Reference	587
11.50.1 Detailed Description	587

11.50.2 Field Documentation	587
11.51 CF_EncoderState Struct Reference	587
11.51.1 Detailed Description	588
11.51.2 Field Documentation	588
11.52 CF_Engine Struct Reference	588
11.52.1 Detailed Description	588
11.52.2 Field Documentation	588
11.53 CF_EotPacket Struct Reference	590
11.53.1 Detailed Description	590
11.53.2 Field Documentation	590
11.54 CF_EotPacket_Payload Struct Reference	590
11.54.1 Detailed Description	591
11.54.2 Field Documentation	591
11.55 CF_Flags_Common Struct Reference	592
11.55.1 Detailed Description	592
11.55.2 Field Documentation	592
11.56 CF_Flags_Rx Struct Reference	593
11.56.1 Detailed Description	594
11.56.2 Field Documentation	594
11.57 CF_Flags_Tx Struct Reference	595
11.57.1 Detailed Description	595
11.57.2 Field Documentation	595
11.58 CF_FreezeCmd Struct Reference	595
11.58.1 Detailed Description	596
11.58.2 Field Documentation	596
11.59 CF_GapComputeArgs_t Struct Reference	596
11.59.1 Detailed Description	596
11.59.2 Field Documentation	596
11.60 CF_GetParam_Payload Struct Reference	597
11.60.1 Detailed Description	597
11.60.2 Field Documentation	597
11.61 CF_GetParamCmd Struct Reference	597
11.61.1 Detailed Description	598
11.61.2 Field Documentation	598
11.62 CF_History Struct Reference	598
11.62.1 Detailed Description	599
11.62.2 Field Documentation	599
11.63 CF_HkChannel_Data Struct Reference	600
11.63.1 Detailed Description	600

11.63.2 Field Documentation	600
11.64 CF_HkCmdCounters Struct Reference	601
11.64.1 Detailed Description	602
11.64.2 Field Documentation	602
11.65 CF_HkCounters Struct Reference	602
11.65.1 Detailed Description	602
11.65.2 Field Documentation	602
11.66 CF_HkFault Struct Reference	603
11.66.1 Detailed Description	604
11.66.2 Field Documentation	604
11.67 CF_HkPacket Struct Reference	605
11.67.1 Detailed Description	605
11.67.2 Field Documentation	606
11.68 CF_HkPacket_Payload Struct Reference	606
11.68.1 Detailed Description	606
11.68.2 Field Documentation	606
11.69 CF_HkRecv Struct Reference	607
11.69.1 Detailed Description	608
11.69.2 Field Documentation	608
11.70 CF_HkSent Struct Reference	608
11.70.1 Detailed Description	609
11.70.2 Field Documentation	609
11.71 CF_Input Struct Reference	609
11.71.1 Detailed Description	610
11.71.2 Field Documentation	610
11.72 CF_Logical_IntHeader Union Reference	610
11.72.1 Detailed Description	610
11.72.2 Field Documentation	611
11.73 CF_Logical_Lv Struct Reference	611
11.73.1 Detailed Description	612
11.73.2 Field Documentation	612
11.74 CF_Logical_PduAck Struct Reference	612
11.74.1 Detailed Description	612
11.74.2 Field Documentation	613
11.75 CF_Logical_PduBuffer Struct Reference	613
11.75.1 Detailed Description	613
11.75.2 Field Documentation	614
11.76 CF_Logical_PduEof Struct Reference	615
11.76.1 Detailed Description	615

11.76.2 Field Documentation	615
11.77 CF_Logical_PduFileDataHeader Struct Reference	615
11.77.1 Detailed Description	616
11.77.2 Field Documentation	616
11.78 CF_Logical_PduFileDirectiveHeader Struct Reference	617
11.78.1 Detailed Description	617
11.78.2 Field Documentation	617
11.79 CF_Logical_PduFin Struct Reference	617
11.79.1 Detailed Description	617
11.79.2 Field Documentation	617
11.80 CF_Logical_PduHeader Struct Reference	618
11.80.1 Detailed Description	619
11.80.2 Field Documentation	619
11.81 CF_Logical_PduMd Struct Reference	621
11.81.1 Detailed Description	621
11.81.2 Field Documentation	621
11.82 CF_Logical_PduNak Struct Reference	622
11.82.1 Detailed Description	622
11.82.2 Field Documentation	622
11.83 CF_Logical_SegmentList Struct Reference	623
11.83.1 Detailed Description	623
11.83.2 Field Documentation	623
11.84 CF_Logical_SegmentRequest Struct Reference	623
11.84.1 Detailed Description	624
11.84.2 Field Documentation	624
11.85 CF_Logical_Tlv Struct Reference	624
11.85.1 Detailed Description	624
11.85.2 Field Documentation	624
11.86 CF_Logical_TlvData Union Reference	625
11.86.1 Detailed Description	625
11.86.2 Field Documentation	625
11.87 CF_Logical_TlvList Struct Reference	626
11.87.1 Detailed Description	626
11.87.2 Field Documentation	626
11.88 CF_NoopCmd Struct Reference	626
11.88.1 Detailed Description	626
11.88.2 Field Documentation	626
11.89 CF_Output Struct Reference	627
11.89.1 Detailed Description	627

11.89.2 Field Documentation	627
11.90 CF_PduCmdMsg Struct Reference	627
11.90.1 Detailed Description	628
11.90.2 Field Documentation	628
11.91 CF_PduTlmMsg Struct Reference	628
11.91.1 Detailed Description	628
11.91.2 Field Documentation	629
11.92 CF_Playback Struct Reference	629
11.92.1 Detailed Description	629
11.92.2 Field Documentation	629
11.93 CF_PlaybackDirCmd Struct Reference	631
11.93.1 Detailed Description	631
11.93.2 Field Documentation	631
11.94 CF_Poll Struct Reference	631
11.94.1 Detailed Description	631
11.94.2 Field Documentation	631
11.95 CF_PollDir Struct Reference	632
11.95.1 Detailed Description	632
11.95.2 Field Documentation	632
11.96 CF_PurgeQueueCmd Struct Reference	633
11.96.1 Detailed Description	634
11.96.2 Field Documentation	634
11.97 CF_ResetCountersCmd Struct Reference	634
11.97.1 Detailed Description	634
11.97.2 Field Documentation	634
11.98 CF_ResumeCmd Struct Reference	635
11.98.1 Detailed Description	635
11.98.2 Field Documentation	635
11.99 CF_RxS2_Data Struct Reference	635
11.99.1 Detailed Description	636
11.99.2 Field Documentation	636
11.100 CF_RxState_Data Struct Reference	636
11.100.1 Detailed Description	637
11.100.2 Field Documentation	637
11.101 CF_SendHkCmd Struct Reference	637
11.101.1 Detailed Description	637
11.101.2 Field Documentation	637
11.102 CF_SetParam_Payload Struct Reference	638
11.102.1 Detailed Description	638

11.102.2 Field Documentation	638
11.103 CF_SetParamCmd Struct Reference	639
11.103.1 Detailed Description	639
11.103.2 Field Documentation	639
11.104 CF_StateData Union Reference	639
11.104.1 Detailed Description	640
11.104.2 Field Documentation	640
11.105 CF_StateFlags Union Reference	640
11.105.1 Detailed Description	640
11.105.2 Field Documentation	640
11.106 CF_SuspendCmd Struct Reference	641
11.106.1 Detailed Description	641
11.106.2 Field Documentation	641
11.107 CF_ThawCmd Struct Reference	642
11.107.1 Detailed Description	642
11.107.2 Field Documentation	642
11.108 CF_Timer Struct Reference	642
11.108.1 Detailed Description	643
11.108.2 Field Documentation	643
11.109 CF_Transaction Struct Reference	643
11.109.1 Detailed Description	644
11.109.2 Field Documentation	644
11.110 CF_Transaction_Payload Struct Reference	647
11.110.1 Detailed Description	647
11.110.2 Field Documentation	647
11.111 CF_Traverse_PriorityArg Struct Reference	648
11.111.1 Detailed Description	648
11.111.2 Field Documentation	648
11.112 CF_Traverse_TransSeqArg Struct Reference	648
11.112.1 Detailed Description	648
11.112.2 Field Documentation	649
11.113 CF_Traverse_WriteHistoryFileArg Struct Reference	649
11.113.1 Detailed Description	649
11.113.2 Field Documentation	649
11.114 CF_Traverse_WriteTxnFileArg Struct Reference	650
11.114.1 Detailed Description	650
11.114.2 Field Documentation	650
11.115 CF_TraverseAll_Arg Struct Reference	651
11.115.1 Detailed Description	651

11.115.2 Field Documentation	651
11.116 CF_TxFile_Payload Struct Reference	651
11.116.1 Detailed Description	652
11.116.2 Field Documentation	652
11.117 CF_TxFileCmd Struct Reference	653
11.117.1 Detailed Description	653
11.117.2 Field Documentation	653
11.118 CF_TxnFilenames Struct Reference	654
11.118.1 Detailed Description	654
11.118.2 Field Documentation	654
11.119 CF_TxS2_Data Struct Reference	654
11.119.1 Detailed Description	654
11.119.2 Field Documentation	654
11.120 CF_TxState_Data Struct Reference	655
11.120.1 Detailed Description	655
11.120.2 Field Documentation	655
11.121 CF_UnionArgs_Payload Union Reference	656
11.121.1 Detailed Description	656
11.121.2 Field Documentation	656
11.122 CF_WakeupCmd Struct Reference	656
11.122.1 Detailed Description	657
11.122.2 Field Documentation	657
11.123 CF_WriteQueue_Payload Struct Reference	657
11.123.1 Detailed Description	657
11.123.2 Field Documentation	657
11.124 CF_WriteQueueCmd Struct Reference	658
11.124.1 Detailed Description	658
11.124.2 Field Documentation	658
11.125 CFE_Config_ArrayValue Struct Reference	659
11.125.1 Detailed Description	659
11.125.2 Field Documentation	659
11.126 CFE_Config_IdNameEntry Struct Reference	659
11.126.1 Detailed Description	659
11.126.2 Field Documentation	659
11.127 CFE_Config_ValueBuffer Union Reference	660
11.127.1 Detailed Description	660
11.127.2 Field Documentation	660
11.128 CFE_Config_ValueEntry Struct Reference	660
11.128.1 Detailed Description	660

11.128.2 Field Documentation	660
11.129 CFE_ES_AppInfo Struct Reference	661
11.129.1 Detailed Description	662
11.129.2 Field Documentation	662
11.130 CFE_ES_AppNameCmd_Payload Struct Reference	665
11.130.1 Detailed Description	665
11.130.2 Field Documentation	665
11.131 CFE_ES_AppReloadCmd_Payload Struct Reference	665
11.131.1 Detailed Description	665
11.131.2 Field Documentation	666
11.132 CFE_ES_BlockStats Struct Reference	666
11.132.1 Detailed Description	666
11.132.2 Field Documentation	666
11.133 CFE_ES_CDSRegDumpRec Struct Reference	667
11.133.1 Detailed Description	667
11.133.2 Field Documentation	667
11.134 CFE_ES_ClearERLogCmd Struct Reference	668
11.134.1 Detailed Description	668
11.134.2 Field Documentation	668
11.135 CFE_ES_ClearSysLogCmd Struct Reference	668
11.135.1 Detailed Description	668
11.135.2 Field Documentation	669
11.136 CFE_ES_DeleteCDSCmd Struct Reference	669
11.136.1 Detailed Description	669
11.136.2 Field Documentation	669
11.137 CFE_ES_DeleteCDSCmd_Payload Struct Reference	669
11.137.1 Detailed Description	670
11.137.2 Field Documentation	670
11.138 CFE_ES_DumpCDSRegistryCmd Struct Reference	670
11.138.1 Detailed Description	670
11.138.2 Field Documentation	670
11.139 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference	670
11.139.1 Detailed Description	671
11.139.2 Field Documentation	671
11.140 CFE_ES_FileNameCmd Struct Reference	671
11.140.1 Detailed Description	671
11.140.2 Field Documentation	671
11.141 CFE_ES_FileNameCmd_Payload Struct Reference	672
11.141.1 Detailed Description	672

11.141.2 Field Documentation	672
11.142 CFE_ES_HousekeepingTlm Struct Reference	672
11.142.1 Detailed Description	672
11.142.2 Field Documentation	672
11.143 CFE_ES_HousekeepingTlm_Payload Struct Reference	673
11.143.1 Detailed Description	675
11.143.2 Field Documentation	675
11.144 CFE_ES_MemPoolStats Struct Reference	681
11.144.1 Detailed Description	681
11.144.2 Field Documentation	681
11.145 CFE_ES_MemStatsTlm Struct Reference	682
11.145.1 Detailed Description	682
11.145.2 Field Documentation	682
11.146 CFE_ES_NoopCmd Struct Reference	683
11.146.1 Detailed Description	683
11.146.2 Field Documentation	683
11.147 CFE_ES_OneAppTlm Struct Reference	683
11.147.1 Detailed Description	683
11.147.2 Field Documentation	683
11.148 CFE_ES_OneAppTlm_Payload Struct Reference	684
11.148.1 Detailed Description	684
11.148.2 Field Documentation	684
11.149 CFE_ES_OverWriteSysLogCmd Struct Reference	684
11.149.1 Detailed Description	684
11.149.2 Field Documentation	684
11.150 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference	685
11.150.1 Detailed Description	685
11.150.2 Field Documentation	685
11.151 CFE_ES_PoolAlign Union Reference	685
11.151.1 Detailed Description	686
11.151.2 Field Documentation	686
11.152 CFE_ES_PoolStatsTlm_Payload Struct Reference	686
11.152.1 Detailed Description	686
11.152.2 Field Documentation	686
11.153 CFE_ES_QueryAllCmd Struct Reference	687
11.153.1 Detailed Description	687
11.153.2 Field Documentation	687
11.154 CFE_ES_QueryAllTasksCmd Struct Reference	687
11.154.1 Detailed Description	687

11.154.2 Field Documentation	688
11.155 CFE_ES_QueryOneCmd Struct Reference	688
11.155.1 Detailed Description	688
11.155.2 Field Documentation	688
11.156 CFE_ES_ReloadAppCmd Struct Reference	688
11.156.1 Detailed Description	689
11.156.2 Field Documentation	689
11.157 CFE_ES_ResetCountersCmd Struct Reference	689
11.157.1 Detailed Description	689
11.157.2 Field Documentation	689
11.158 CFE_ES_ResetPRCountCmd Struct Reference	689
11.158.1 Detailed Description	690
11.158.2 Field Documentation	690
11.159 CFE_ES_RestartAppCmd Struct Reference	690
11.159.1 Detailed Description	690
11.159.2 Field Documentation	690
11.160 CFE_ES_RestartCmd Struct Reference	690
11.160.1 Detailed Description	691
11.160.2 Field Documentation	691
11.161 CFE_ES_RestartCmd_Payload Struct Reference	691
11.161.1 Detailed Description	691
11.161.2 Field Documentation	691
11.162 CFE_ES_SendHkCmd Struct Reference	692
11.162.1 Detailed Description	692
11.162.2 Field Documentation	692
11.163 CFE_ES_SendMemPoolStatsCmd Struct Reference	692
11.163.1 Detailed Description	692
11.163.2 Field Documentation	692
11.164 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference	693
11.164.1 Detailed Description	693
11.164.2 Field Documentation	693
11.165 CFE_ES_SetMaxPRCountCmd Struct Reference	693
11.165.1 Detailed Description	693
11.165.2 Field Documentation	694
11.166 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference	694
11.166.1 Detailed Description	694
11.166.2 Field Documentation	694
11.167 CFE_ES_SetPerfFilterMaskCmd Struct Reference	694
11.167.1 Detailed Description	695

11.167.2 Field Documentation	695
11.168 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference	695
11.168.1 Detailed Description	695
11.168.2 Field Documentation	695
11.169 CFE_ES_SetPerfTriggerMaskCmd Struct Reference	696
11.169.1 Detailed Description	696
11.169.2 Field Documentation	696
11.170 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference	696
11.170.1 Detailed Description	696
11.170.2 Field Documentation	696
11.171 CFE_ES_StartApp Struct Reference	697
11.171.1 Detailed Description	697
11.171.2 Field Documentation	697
11.172 CFE_ES_StartAppCmd_Payload Struct Reference	697
11.172.1 Detailed Description	698
11.172.2 Field Documentation	698
11.173 CFE_ES_StartPerfCmd_Payload Struct Reference	699
11.173.1 Detailed Description	699
11.173.2 Field Documentation	699
11.174 CFE_ES_StartPerfDataCmd Struct Reference	699
11.174.1 Detailed Description	699
11.174.2 Field Documentation	699
11.175 CFE_ES_StopAppCmd Struct Reference	700
11.175.1 Detailed Description	700
11.175.2 Field Documentation	700
11.176 CFE_ES_StopPerfCmd_Payload Struct Reference	700
11.176.1 Detailed Description	701
11.176.2 Field Documentation	701
11.177 CFE_ES_StopPerfDataCmd Struct Reference	701
11.177.1 Detailed Description	701
11.177.2 Field Documentation	701
11.178 CFE_ES_TaskInfo Struct Reference	701
11.178.1 Detailed Description	702
11.178.2 Field Documentation	702
11.179 CFE_ES_WriteERLogCmd Struct Reference	703
11.179.1 Detailed Description	703
11.179.2 Field Documentation	703
11.180 CFE_ES_WriteSysLogCmd Struct Reference	704
11.180.1 Detailed Description	704

11.180.2 Field Documentation	704
11.181 CFE_EVS_AddEventFilterCmd Struct Reference	704
11.181.1 Detailed Description	704
11.181.2 Field Documentation	704
11.182 CFE_EVS_AppDataCmd_Payload Struct Reference	705
11.182.1 Detailed Description	705
11.182.2 Field Documentation	705
11.183 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference	705
11.183.1 Detailed Description	705
11.183.2 Field Documentation	706
11.184 CFE_EVS_AppNameCmd_Payload Struct Reference	706
11.184.1 Detailed Description	706
11.184.2 Field Documentation	706
11.185 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference	706
11.185.1 Detailed Description	707
11.185.2 Field Documentation	707
11.186 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference	707
11.186.1 Detailed Description	707
11.186.2 Field Documentation	707
11.187 CFE_EVS_AppTlmData Struct Reference	708
11.187.1 Detailed Description	708
11.187.2 Field Documentation	708
11.188 CFE_EVS_BinFilter Struct Reference	709
11.188.1 Detailed Description	709
11.188.2 Field Documentation	709
11.189 CFE_EVS_BitMaskCmd_Payload Struct Reference	709
11.189.1 Detailed Description	710
11.189.2 Field Documentation	710
11.190 CFE_EVS_ClearLogCmd Struct Reference	710
11.190.1 Detailed Description	710
11.190.2 Field Documentation	710
11.191 CFE_EVS_DeleteEventFilterCmd Struct Reference	711
11.191.1 Detailed Description	711
11.191.2 Field Documentation	711
11.192 CFE_EVS_DisableAppEventsCmd Struct Reference	711
11.192.1 Detailed Description	711
11.192.2 Field Documentation	711
11.193 CFE_EVS_DisableAppEventTypeCmd Struct Reference	712
11.193.1 Detailed Description	712

11.193.2 Field Documentation	712
11.194 CFE_EVS_DisableEventTypeCmd Struct Reference	712
11.194.1 Detailed Description	712
11.194.2 Field Documentation	713
11.195 CFE_EVS_DisablePortsCmd Struct Reference	713
11.195.1 Detailed Description	713
11.195.2 Field Documentation	713
11.196 CFE_EVS_EnableAppEventsCmd Struct Reference	713
11.196.1 Detailed Description	714
11.196.2 Field Documentation	714
11.197 CFE_EVS_EnableAppEventCmd Struct Reference	714
11.197.1 Detailed Description	714
11.197.2 Field Documentation	714
11.198 CFE_EVS_EnableEventTypeCmd Struct Reference	715
11.198.1 Detailed Description	715
11.198.2 Field Documentation	715
11.199 CFE_EVS_EnablePortsCmd Struct Reference	715
11.199.1 Detailed Description	715
11.199.2 Field Documentation	716
11.200 CFE_EVS_HousekeepingTlm Struct Reference	716
11.200.1 Detailed Description	716
11.200.2 Field Documentation	716
11.201 CFE_EVS_HousekeepingTlm_Payload Struct Reference	716
11.201.1 Detailed Description	717
11.201.2 Field Documentation	717
11.202 CFE_EVS_LogFileCmd_Payload Struct Reference	720
11.202.1 Detailed Description	720
11.202.2 Field Documentation	720
11.203 CFE_EVS_LongEventTlm Struct Reference	720
11.203.1 Detailed Description	720
11.203.2 Field Documentation	720
11.204 CFE_EVS_LongEventTlm_Payload Struct Reference	721
11.204.1 Detailed Description	721
11.204.2 Field Documentation	721
11.205 CFE_EVS_NoopCmd Struct Reference	722
11.205.1 Detailed Description	722
11.205.2 Field Documentation	722
11.206 CFE_EVS_PacketID Struct Reference	722
11.206.1 Detailed Description	722

11.206.2 Field Documentation	723
11.207 CFE_EVS_ResetAllFiltersCmd Struct Reference	723
11.207.1 Detailed Description	724
11.207.2 Field Documentation	724
11.208 CFE_EVS_ResetAppCounterCmd Struct Reference	724
11.208.1 Detailed Description	724
11.208.2 Field Documentation	724
11.209 CFE_EVS_ResetCountersCmd Struct Reference	724
11.209.1 Detailed Description	725
11.209.2 Field Documentation	725
11.210 CFE_EVS_ResetFilterCmd Struct Reference	725
11.210.1 Detailed Description	725
11.210.2 Field Documentation	725
11.211 CFE_EVS_SendHkCmd Struct Reference	725
11.211.1 Detailed Description	726
11.211.2 Field Documentation	726
11.212 CFE_EVS_SetEventFormatCode_Payload Struct Reference	726
11.212.1 Detailed Description	726
11.212.2 Field Documentation	726
11.213 CFE_EVS_SetEventFormatModeCmd Struct Reference	727
11.213.1 Detailed Description	727
11.213.2 Field Documentation	727
11.214 CFE_EVS_SetFilterCmd Struct Reference	727
11.214.1 Detailed Description	727
11.214.2 Field Documentation	727
11.215 CFE_EVS_SetLogMode_Payload Struct Reference	728
11.215.1 Detailed Description	728
11.215.2 Field Documentation	728
11.216 CFE_EVS_SetLogModeCmd Struct Reference	728
11.216.1 Detailed Description	729
11.216.2 Field Documentation	729
11.217 CFE_EVS_ShortEventTlm Struct Reference	729
11.217.1 Detailed Description	729
11.217.2 Field Documentation	729
11.218 CFE_EVS_ShortEventTlm_Payload Struct Reference	730
11.218.1 Detailed Description	730
11.218.2 Field Documentation	730
11.219 CFE_EVS_WriteAppDataFileCmd Struct Reference	730
11.219.1 Detailed Description	730

11.219.2 Field Documentation	730
11.220 CFE_EVS_WriteLogFileCmd Struct Reference	731
11.220.1 Detailed Description	731
11.220.2 Field Documentation	731
11.221 CFE_FS_FileWriteMetaData Struct Reference	731
11.221.1 Detailed Description	732
11.221.2 Field Documentation	732
11.222 CFE_FS_Header Struct Reference	733
11.222.1 Detailed Description	733
11.222.2 Field Documentation	733
11.223 CFE_SB_AllSubscriptionsTlm Struct Reference	734
11.223.1 Detailed Description	734
11.223.2 Field Documentation	734
11.224 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference	735
11.224.1 Detailed Description	735
11.224.2 Field Documentation	735
11.225 CFE_SB_DisableRouteCmd Struct Reference	736
11.225.1 Detailed Description	736
11.225.2 Field Documentation	736
11.226 CFE_SB_DisableSubReportingCmd Struct Reference	736
11.226.1 Detailed Description	736
11.226.2 Field Documentation	737
11.227 CFE_SB_EnableRouteCmd Struct Reference	737
11.227.1 Detailed Description	737
11.227.2 Field Documentation	737
11.228 CFE_SB_EnableSubReportingCmd Struct Reference	737
11.228.1 Detailed Description	737
11.228.2 Field Documentation	738
11.229 CFE_SB_HousekeepingTlm Struct Reference	738
11.229.1 Detailed Description	738
11.229.2 Field Documentation	738
11.230 CFE_SB_HousekeepingTlm_Payload Struct Reference	738
11.230.1 Detailed Description	739
11.230.2 Field Documentation	739
11.231 CFE_SB_Msg Union Reference	742
11.231.1 Detailed Description	742
11.231.2 Field Documentation	742
11.232 CFE_SB_MsgId_t Struct Reference	743
11.232.1 Detailed Description	743

11.232.2 Field Documentation	743
11.233 CFE_SB_MsgMapFileEntry Struct Reference	743
11.233.1 Detailed Description	743
11.233.2 Field Documentation	743
11.234 CFE_SB_NoopCmd Struct Reference	744
11.234.1 Detailed Description	744
11.234.2 Field Documentation	744
11.235 CFE_SB_PipeDepthStats Struct Reference	744
11.235.1 Detailed Description	745
11.235.2 Field Documentation	745
11.236 CFE_SB_PipeInfoEntry Struct Reference	745
11.236.1 Detailed Description	746
11.236.2 Field Documentation	746
11.237 CFE_SB_Qos_t Struct Reference	747
11.237.1 Detailed Description	747
11.237.2 Field Documentation	747
11.238 CFE_SB_ResetCountersCmd Struct Reference	748
11.238.1 Detailed Description	748
11.238.2 Field Documentation	748
11.239 CFE_SB_RouteCmd_Payload Struct Reference	748
11.239.1 Detailed Description	748
11.239.2 Field Documentation	748
11.240 CFE_SB_RoutingFileEntry Struct Reference	749
11.240.1 Detailed Description	749
11.240.2 Field Documentation	749
11.241 CFE_SB_SendHkCmd Struct Reference	750
11.241.1 Detailed Description	750
11.241.2 Field Documentation	750
11.242 CFE_SB_SendPrevSubsCmd Struct Reference	750
11.242.1 Detailed Description	751
11.242.2 Field Documentation	751
11.243 CFE_SB_SendSbStatsCmd Struct Reference	751
11.243.1 Detailed Description	751
11.243.2 Field Documentation	751
11.244 CFE_SB_SingleSubscriptionTlm Struct Reference	751
11.244.1 Detailed Description	751
11.244.2 Field Documentation	752
11.245 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference	752
11.245.1 Detailed Description	752

11.245.2 Field Documentation	752
11.246 CFE_SB_StatsTlm Struct Reference	753
11.246.1 Detailed Description	753
11.246.2 Field Documentation	753
11.247 CFE_SB_StatsTlm_Payload Struct Reference	753
11.247.1 Detailed Description	754
11.247.2 Field Documentation	754
11.248 CFE_SB_SubEntries Struct Reference	757
11.248.1 Detailed Description	757
11.248.2 Field Documentation	757
11.249 CFE_SB_WriteFileInfoCmd_Payload Struct Reference	758
11.249.1 Detailed Description	758
11.249.2 Field Documentation	758
11.250 CFE_SB_WriteMapInfoCmd Struct Reference	758
11.250.1 Detailed Description	758
11.250.2 Field Documentation	758
11.251 CFE_SB_WritePipeInfoCmd Struct Reference	759
11.251.1 Detailed Description	759
11.251.2 Field Documentation	759
11.252 CFE_SB_WriteRoutingInfoCmd Struct Reference	759
11.252.1 Detailed Description	759
11.252.2 Field Documentation	760
11.253 CFE_TBL_AbortLoadCmd Struct Reference	760
11.253.1 Detailed Description	760
11.253.2 Field Documentation	760
11.254 CFE_TBL_AbortLoadCmd_Payload Struct Reference	760
11.254.1 Detailed Description	761
11.254.2 Field Documentation	761
11.255 CFE_TBL_ActivateCmd Struct Reference	761
11.255.1 Detailed Description	761
11.255.2 Field Documentation	761
11.256 CFE_TBL_ActivateCmd_Payload Struct Reference	762
11.256.1 Detailed Description	762
11.256.2 Field Documentation	762
11.257 CFE_TBL_DelCDSCmd_Payload Struct Reference	762
11.257.1 Detailed Description	762
11.257.2 Field Documentation	762
11.258 CFE_TBL_DeleteCDSCmd Struct Reference	763
11.258.1 Detailed Description	763

11.258.2 Field Documentation	763
11.259 CFE_TBL_DumpCmd Struct Reference	763
11.259.1 Detailed Description	763
11.259.2 Field Documentation	763
11.260 CFE_TBL_DumpCmd_Payload Struct Reference	764
11.260.1 Detailed Description	764
11.260.2 Field Documentation	764
11.261 CFE_TBL_DumpRegistryCmd Struct Reference	765
11.261.1 Detailed Description	765
11.261.2 Field Documentation	765
11.262 CFE_TBL_DumpRegistryCmd_Payload Struct Reference	765
11.262.1 Detailed Description	765
11.262.2 Field Documentation	766
11.263 CFE_TBL_File_Hdr Struct Reference	766
11.263.1 Detailed Description	766
11.263.2 Field Documentation	766
11.264 CFE_TBL_FileDef Struct Reference	767
11.264.1 Detailed Description	767
11.264.2 Field Documentation	767
11.265 CFE_TBL_HousekeepingTlm Struct Reference	768
11.265.1 Detailed Description	768
11.265.2 Field Documentation	768
11.266 CFE_TBL_HousekeepingTlm_Payload Struct Reference	769
11.266.1 Detailed Description	770
11.266.2 Field Documentation	770
11.267 CFE_TBL_Info Struct Reference	773
11.267.1 Detailed Description	773
11.267.2 Field Documentation	773
11.268 CFE_TBL_LoadCmd Struct Reference	775
11.268.1 Detailed Description	775
11.268.2 Field Documentation	775
11.269 CFE_TBL_LoadCmd_Payload Struct Reference	775
11.269.1 Detailed Description	775
11.269.2 Field Documentation	775
11.270 CFE_TBL_NoopCmd Struct Reference	776
11.270.1 Detailed Description	776
11.270.2 Field Documentation	776
11.271 CFE_TBL_NotifyCmd Struct Reference	776
11.271.1 Detailed Description	776

11.271.2 Field Documentation	776
11.272 CFE_TBL_NotifyCmd_Payload Struct Reference	777
11.272.1 Detailed Description	777
11.272.2 Field Documentation	777
11.273 CFE_TBL_ResetCountersCmd Struct Reference	777
11.273.1 Detailed Description	777
11.273.2 Field Documentation	777
11.274 CFE_TBL_SendHkCmd Struct Reference	778
11.274.1 Detailed Description	778
11.274.2 Field Documentation	778
11.275 CFE_TBL_SendRegistryCmd Struct Reference	778
11.275.1 Detailed Description	778
11.275.2 Field Documentation	778
11.276 CFE_TBL_SendRegistryCmd_Payload Struct Reference	779
11.276.1 Detailed Description	779
11.276.2 Field Documentation	779
11.277 CFE_TBL_TableRegistryTlm Struct Reference	779
11.277.1 Detailed Description	779
11.277.2 Field Documentation	780
11.278 CFE_TBL_TblRegPacket_Payload Struct Reference	780
11.278.1 Detailed Description	781
11.278.2 Field Documentation	781
11.279 CFE_TBL_ValidateCmd Struct Reference	783
11.279.1 Detailed Description	783
11.279.2 Field Documentation	783
11.280 CFE_TBL_ValidateCmd_Payload Struct Reference	784
11.280.1 Detailed Description	784
11.280.2 Field Documentation	784
11.281 CFE_TIME_AddAdjustCmd Struct Reference	784
11.281.1 Detailed Description	785
11.281.2 Field Documentation	785
11.282 CFE_TIME_AddDelayCmd Struct Reference	785
11.282.1 Detailed Description	785
11.282.2 Field Documentation	785
11.283 CFE_TIME_AddOneHzAdjustmentCmd Struct Reference	786
11.283.1 Detailed Description	786
11.283.2 Field Documentation	786
11.284 CFE_TIME_DiagnosticTlm Struct Reference	786
11.284.1 Detailed Description	786

11.284.2 Field Documentation	787
11.285 CFE_TIME_DiagnosticTlm_Payload Struct Reference	787
11.285.1 Detailed Description	789
11.285.2 Field Documentation	789
11.286 CFE_TIME_FakeToneCmd Struct Reference	795
11.286.1 Detailed Description	795
11.286.2 Field Documentation	795
11.287 CFE_TIME_HousekeepingTlm Struct Reference	795
11.287.1 Detailed Description	796
11.287.2 Field Documentation	796
11.288 CFE_TIME_HousekeepingTlm_Payload Struct Reference	796
11.288.1 Detailed Description	797
11.288.2 Field Documentation	797
11.289 CFE_TIME_LeapsCmd_Payload Struct Reference	799
11.289.1 Detailed Description	799
11.289.2 Field Documentation	799
11.290 CFE_TIME_NoopCmd Struct Reference	799
11.290.1 Detailed Description	799
11.290.2 Field Documentation	800
11.291 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference	800
11.291.1 Detailed Description	800
11.291.2 Field Documentation	800
11.292 CFE_TIME_OneHzCmd Struct Reference	800
11.292.1 Detailed Description	800
11.292.2 Field Documentation	800
11.293 CFE_TIME_ResetCountersCmd Struct Reference	801
11.293.1 Detailed Description	801
11.293.2 Field Documentation	801
11.294 CFE_TIME_SendDiagnosticCmd Struct Reference	801
11.294.1 Detailed Description	801
11.294.2 Field Documentation	801
11.295 CFE_TIME_SendHkCmd Struct Reference	802
11.295.1 Detailed Description	802
11.295.2 Field Documentation	802
11.296 CFE_TIME_SetLeapSecondsCmd Struct Reference	802
11.296.1 Detailed Description	802
11.296.2 Field Documentation	802
11.297 CFE_TIME_SetMETCmd Struct Reference	803
11.297.1 Detailed Description	803

11.297.2 Field Documentation	803
11.298 CFE_TIME_SetSignalCmd Struct Reference	803
11.298.1 Detailed Description	803
11.298.2 Field Documentation	803
11.299 CFE_TIME_SetSourceCmd Struct Reference	804
11.299.1 Detailed Description	804
11.299.2 Field Documentation	804
11.300 CFE_TIME_SetStateCmd Struct Reference	804
11.300.1 Detailed Description	805
11.300.2 Field Documentation	805
11.301 CFE_TIME_SetSTCFCmd Struct Reference	805
11.301.1 Detailed Description	805
11.301.2 Field Documentation	805
11.302 CFE_TIME_SetTimeCmd Struct Reference	806
11.302.1 Detailed Description	806
11.302.2 Field Documentation	806
11.303 CFE_TIME_SignalCmd_Payload Struct Reference	806
11.303.1 Detailed Description	806
11.303.2 Field Documentation	807
11.304 CFE_TIME_SourceCmd_Payload Struct Reference	807
11.304.1 Detailed Description	807
11.304.2 Field Documentation	807
11.305 CFE_TIME_StateCmd_Payload Struct Reference	807
11.305.1 Detailed Description	808
11.305.2 Field Documentation	808
11.306 CFE_TIME_SubAdjustCmd Struct Reference	808
11.306.1 Detailed Description	808
11.306.2 Field Documentation	808
11.307 CFE_TIME_SubDelayCmd Struct Reference	808
11.307.1 Detailed Description	809
11.307.2 Field Documentation	809
11.308 CFE_TIME_SubOneHzAdjustmentCmd Struct Reference	809
11.308.1 Detailed Description	809
11.308.2 Field Documentation	809
11.309 CFE_TIME_SysTime Struct Reference	810
11.309.1 Detailed Description	810
11.309.2 Field Documentation	810
11.310 CFE_TIME_TimeCmd_Payload Struct Reference	810
11.310.1 Detailed Description	811

11.310.2 Field Documentation	811
11.311 CFE_TIME_ToneDataCmd Struct Reference	811
11.311.1 Detailed Description	811
11.311.2 Field Documentation	811
11.312 CFE_TIME_ToneDataCmd_Payload Struct Reference	812
11.312.1 Detailed Description	812
11.312.2 Field Documentation	812
11.313 CFE_TIME_ToneSignalCmd Struct Reference	812
11.313.1 Detailed Description	813
11.313.2 Field Documentation	813
11.314 OS_bin_sem_prop_t Struct Reference	813
11.314.1 Detailed Description	813
11.314.2 Field Documentation	813
11.315 OS_condvar_prop_t Struct Reference	814
11.315.1 Detailed Description	814
11.315.2 Field Documentation	814
11.316 OS_count_sem_prop_t Struct Reference	814
11.316.1 Detailed Description	814
11.316.2 Field Documentation	814
11.317 os_dirent_t Struct Reference	815
11.317.1 Detailed Description	815
11.317.2 Field Documentation	815
11.318 OS_FdSet Struct Reference	815
11.318.1 Detailed Description	815
11.318.2 Field Documentation	816
11.319 OS_file_prop_t Struct Reference	816
11.319.1 Detailed Description	816
11.319.2 Field Documentation	816
11.320 os_fsinfo_t Struct Reference	816
11.320.1 Detailed Description	817
11.320.2 Field Documentation	817
11.321 os_fstat_t Struct Reference	817
11.321.1 Detailed Description	817
11.321.2 Field Documentation	818
11.322 OS_heap_prop_t Struct Reference	818
11.322.1 Detailed Description	818
11.322.2 Field Documentation	818
11.323 OS_module_address_t Struct Reference	819
11.323.1 Detailed Description	819

11.323.2 Field Documentation	819
11.324 OS_module_prop_t Struct Reference	820
11.324.1 Detailed Description	820
11.324.2 Field Documentation	820
11.325 OS_mut_sem_prop_t Struct Reference	821
11.325.1 Detailed Description	821
11.325.2 Field Documentation	821
11.326 OS_queue_prop_t Struct Reference	821
11.326.1 Detailed Description	821
11.326.2 Field Documentation	821
11.327 OS_SockAddr_t Struct Reference	822
11.327.1 Detailed Description	822
11.327.2 Field Documentation	822
11.328 OS_SockAddrData_t Union Reference	822
11.328.1 Detailed Description	823
11.328.2 Field Documentation	823
11.329 OS_socket_prop_t Struct Reference	823
11.329.1 Detailed Description	823
11.329.2 Field Documentation	824
11.330 OS_static_symbol_record_t Struct Reference	824
11.330.1 Detailed Description	824
11.330.2 Field Documentation	824
11.331 OS_statvfs_t Struct Reference	825
11.331.1 Detailed Description	825
11.331.2 Field Documentation	825
11.332 OS_task_prop_t Struct Reference	825
11.332.1 Detailed Description	825
11.332.2 Field Documentation	825
11.333 OS_time_t Struct Reference	826
11.333.1 Detailed Description	826
11.333.2 Field Documentation	826
11.334 OS_timebase_prop_t Struct Reference	827
11.334.1 Detailed Description	827
11.334.2 Field Documentation	827
11.335 OS_timer_prop_t Struct Reference	827
11.335.1 Detailed Description	828
11.335.2 Field Documentation	828
12 File Documentation	828

12.1 apps/cf/config/default_cf_extern_typedefs.h File Reference	828
12.1.1 Detailed Description	829
12.1.2 Typedef Documentation	829
12.1.3 Enumeration Type Documentation	830
12.2 apps/cf/config/default_cf_fcncodes.h File Reference	831
12.2.1 Detailed Description	831
12.3 apps/cf/config/default_cf_interface_cfg.h File Reference	831
12.3.1 Detailed Description	832
12.4 apps/cf/config/default_cf_internal_cfg.h File Reference	832
12.4.1 Detailed Description	832
12.5 apps/cf/config/default_cf_mission_cfg.h File Reference	833
12.5.1 Detailed Description	833
12.5.2 Macro Definition Documentation	833
12.6 apps/cf/config/default_cf_msg.h File Reference	833
12.6.1 Detailed Description	833
12.6.2 Macro Definition Documentation	834
12.7 apps/cf/config/default_cf_msgdefs.h File Reference	834
12.7.1 Detailed Description	836
12.8 apps/cf/config/default_cf_msgids.h File Reference	836
12.8.1 Detailed Description	836
12.9 apps/cf/config/default_cf_msgstruct.h File Reference	836
12.9.1 Detailed Description	838
12.10 apps/cf/config/default_cf_platform_cfg.h File Reference	839
12.10.1 Detailed Description	839
12.10.2 Macro Definition Documentation	839
12.11 apps/cf/config/default_cf_tbl.h File Reference	840
12.11.1 Detailed Description	840
12.12 apps/cf/config/default_cf_tbldefs.h File Reference	840
12.12.1 Detailed Description	840
12.12.2 Typedef Documentation	840
12.13 apps/cf/config/default_cf_tblstruct.h File Reference	841
12.13.1 Detailed Description	841
12.13.2 Typedef Documentation	841
12.14 apps/cf/config/default_cf_topicids.h File Reference	841
12.14.1 Detailed Description	842
12.14.2 Macro Definition Documentation	842
12.15 apps/cf/docs/dox_src/cfs_cf.dox File Reference	843
12.16 apps/cf/fsw/inc(cf_events.h File Reference	843
12.16.1 Detailed Description	848

12.17 apps/cf/fsw/inc(cf_perfids.h File Reference)	848
12.17.1 Detailed Description	849
12.18 apps/cf/fsw/src(cf_app.c File Reference)	849
12.18.1 Detailed Description	850
12.18.2 Function Documentation	850
12.18.3 Variable Documentation	855
12.19 apps/cf/fsw/src(cf_app.h File Reference)	855
12.19.1 Detailed Description	856
12.19.2 Macro Definition Documentation	856
12.19.3 Function Documentation	857
12.19.4 Variable Documentation	862
12.20 apps/cf/fsw/src(cf_assert.h File Reference)	862
12.20.1 Detailed Description	863
12.20.2 Macro Definition Documentation	863
12.21 apps/cf/fsw/src(cf_cfdp.c File Reference)	863
12.21.1 Detailed Description	865
12.21.2 Function Documentation	866
12.22 apps/cf/fsw/src(cf_cfdp.h File Reference)	907
12.22.1 Detailed Description	909
12.22.2 Typedef Documentation	910
12.22.3 Function Documentation	910
12.23 apps/cf/fsw/src(cf_cfdp_dispatch.c File Reference)	947
12.23.1 Function Documentation	948
12.24 apps/cf/fsw/src(cf_cfdp_dispatch.h File Reference)	951
12.24.1 Detailed Description	951
12.24.2 Typedef Documentation	951
12.24.3 Function Documentation	952
12.25 apps/cf/fsw/src(cf_cfdp_pdu.h File Reference)	955
12.25.1 Detailed Description	957
12.25.2 Macro Definition Documentation	957
12.25.3 Typedef Documentation	958
12.25.4 Enumeration Type Documentation	959
12.26 apps/cf/fsw/src(cf_cfdp_r.c File Reference)	961
12.26.1 Detailed Description	963
12.26.2 Function Documentation	963
12.27 apps/cf/fsw/src(cf_cfdp_r.h File Reference)	985
12.27.1 Detailed Description	986
12.27.2 Function Documentation	987
12.28 apps/cf/fsw/src(cf_cfdp_s.c File Reference)	1008

12.28.1 Detailed Description	1010
12.28.2 Function Documentation	1010
12.29 apps/cf/fsw/src/cf_cfdp_s.h File Reference	1029
12.29.1 Detailed Description	1030
12.29.2 Function Documentation	1030
12.30 apps/cf/fsw/src/cf_cfdp_sbintf.c File Reference	1049
12.30.1 Detailed Description	1050
12.30.2 Function Documentation	1050
12.31 apps/cf/fsw/src/cf_cfdp_sbintf.h File Reference	1053
12.31.1 Detailed Description	1054
12.31.2 Typedef Documentation	1054
12.31.3 Function Documentation	1054
12.32 apps/cf/fsw/src/cf_cfdp_types.h File Reference	1058
12.32.1 Detailed Description	1061
12.32.2 Macro Definition Documentation	1061
12.32.3 Typedef Documentation	1062
12.32.4 Enumeration Type Documentation	1063
12.33 apps/cf/fsw/src/cf_chunk.c File Reference	1065
12.33.1 Detailed Description	1066
12.33.2 Function Documentation	1066
12.34 apps/cf/fsw/src/cf_chunk.h File Reference	1075
12.34.1 Detailed Description	1076
12.34.2 Typedef Documentation	1076
12.34.3 Function Documentation	1077
12.35 apps/cf/fsw/src/cf_clist.c File Reference	1086
12.35.1 Detailed Description	1086
12.35.2 Function Documentation	1086
12.36 apps/cf/fsw/src/cf_clist.h File Reference	1091
12.36.1 Detailed Description	1092
12.36.2 Macro Definition Documentation	1092
12.36.3 Typedef Documentation	1092
12.36.4 Enumeration Type Documentation	1093
12.36.5 Function Documentation	1093
12.37 apps/cf/fsw/src/cf_cmd.c File Reference	1097
12.37.1 Detailed Description	1099
12.37.2 Function Documentation	1099
12.38 apps/cf/fsw/src/cf_cmd.h File Reference	1129
12.38.1 Detailed Description	1131
12.38.2 Typedef Documentation	1131

12.38.3 Enumeration Type Documentation	1132
12.38.4 Function Documentation	1132
12.39 apps/cf/fsw/src(cf_codec.c File Reference	1162
12.39.1 Detailed Description	1165
12.39.2 Macro Definition Documentation	1165
12.39.3 Typedef Documentation	1165
12.39.4 Function Documentation	1165
12.39.5 Variable Documentation	1189
12.40 apps/cf/fsw/src(cf_codec.h File Reference	1191
12.40.1 Detailed Description	1194
12.40.2 Macro Definition Documentation	1194
12.40.3 Typedef Documentation	1196
12.40.4 Function Documentation	1196
12.41 apps/cf/fsw/src(cf_crc.c File Reference	1220
12.41.1 Detailed Description	1220
12.41.2 Function Documentation	1221
12.42 apps/cf/fsw/src(cf_crc.h File Reference	1222
12.42.1 Detailed Description	1222
12.42.2 Typedef Documentation	1222
12.42.3 Function Documentation	1222
12.43 apps/cf/fsw/src(cf_dispatch.c File Reference	1224
12.43.1 Detailed Description	1224
12.43.2 Function Documentation	1224
12.44 apps/cf/fsw/src(cf_dispatch.h File Reference	1226
12.44.1 Detailed Description	1226
12.44.2 Function Documentation	1226
12.45 apps/cf/fsw/src(cf_eds_dispatch.c File Reference	1228
12.45.1 Function Documentation	1229
12.45.2 Variable Documentation	1230
12.46 apps/cf/fsw/src(cf_logical_pdu.h File Reference	1231
12.46.1 Detailed Description	1232
12.46.2 Macro Definition Documentation	1232
12.46.3 Typedef Documentation	1233
12.47 apps/cf/fsw/src(cf_timer.c File Reference	1235
12.47.1 Detailed Description	1235
12.47.2 Function Documentation	1235
12.48 apps/cf/fsw/src(cf_timer.h File Reference	1237
12.48.1 Detailed Description	1237
12.48.2 Typedef Documentation	1238

12.48.3 Function Documentation	1238
12.49 apps/cf/fsw/src(cf_utils.c File Reference	1240
12.49.1 Detailed Description	1241
12.49.2 Function Documentation	1241
12.50 apps/cf/fsw/src(cf_utils.h File Reference	1258
12.50.1 Detailed Description	1260
12.50.2 Typedef Documentation	1260
12.50.3 Function Documentation	1261
12.51 apps/cf/fsw/src(cf_verify.h File Reference	1280
12.51.1 Detailed Description	1280
12.52 apps/cf/fsw/src(cf_version.h File Reference	1280
12.52.1 Detailed Description	1281
12.53 apps/cf/fsw/tables(cf_def_config.c File Reference	1281
12.53.1 Detailed Description	1281
12.53.2 Variable Documentation	1281
12.54 build/osal_public_api/inc/osconfig.h File Reference	1281
12.54.1 Macro Definition Documentation	1283
12.55 cfe/cmake/sample_defs/example_mission_cfg.h File Reference	1287
12.55.1 Detailed Description	1288
12.55.2 Macro Definition Documentation	1288
12.56 cfe/cmake/sample_defs/example_platform_cfg.h File Reference	1298
12.56.1 Detailed Description	1302
12.56.2 Macro Definition Documentation	1302
12.57 cfe/cmake/sample_defs/sample_perfids.h File Reference	1342
12.57.1 Detailed Description	1343
12.57.2 Macro Definition Documentation	1343
12.58 cfe/docs/src(cfe_api.dox File Reference	1344
12.59 cfe/docs/src(cfe_es.dox File Reference	1344
12.60 cfe/docs/src(cfe_evs.dox File Reference	1344
12.61 cfe/docs/src(cfe_frontpage.dox File Reference	1344
12.62 cfe/docs/src(cfe_glossary.dox File Reference	1344
12.63 cfe/docs/src(cfe_sb.dox File Reference	1344
12.64 cfe/docs/src(cfe_tbl.dox File Reference	1344
12.65 cfe/docs/src(cfe_time.dox File Reference	1344
12.66 cfe/docs/src(cfe_xref.dox File Reference	1344
12.67 cfe/docs/src(cfs_versions.dox File Reference	1344
12.68 cfe/modules/config/fsw/inc(cfe_config_external.h File Reference	1344
12.68.1 Detailed Description	1345
12.68.2 Function Documentation	1345

12.69 cfe/modules/config/fsw/inc/cfe_config_init.h File Reference	1345
12.69.1 Detailed Description	1345
12.69.2 Function Documentation	1345
12.70 cfe/modules/config/fsw/inc/cfe_config_lookup.h File Reference	1345
12.70.1 Detailed Description	1345
12.70.2 Function Documentation	1346
12.71 cfe/modules/config/fsw/inc/cfe_config_nametable.h File Reference	1346
12.71.1 Detailed Description	1346
12.71.2 Typedef Documentation	1346
12.71.3 Variable Documentation	1346
12.72 cfe/modules/config/fsw/inc/cfe_config_set.h File Reference	1346
12.72.1 Detailed Description	1347
12.72.2 Function Documentation	1347
12.73 cfe/modules/config/fsw/inc/cfe_config_table.h File Reference	1347
12.73.1 Detailed Description	1347
12.73.2 Typedef Documentation	1348
12.73.3 Enumeration Type Documentation	1348
12.74 cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h File Reference	1348
12.74.1 Detailed Description	1349
12.74.2 Macro Definition Documentation	1349
12.75 cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h File Reference	1350
12.75.1 Detailed Description	1350
12.75.2 Macro Definition Documentation	1350
12.76 cfe/modules/core_api/config/default_cfe_mission_cfg.h File Reference	1352
12.76.1 Detailed Description	1352
12.77 cfe/modules/core_api/config/default_cfe_msgids.h File Reference	1353
12.77.1 Detailed Description	1353
12.78 cfe/modules/core_api/fsw/inc/cfe.h File Reference	1353
12.78.1 Detailed Description	1353
12.79 cfe/modules/core_api/fsw/inc/cfe_config.h File Reference	1353
12.79.1 Detailed Description	1354
12.79.2 Function Documentation	1354
12.80 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference	1357
12.80.1 Detailed Description	1358
12.80.2 Macro Definition Documentation	1358
12.80.3 Typedef Documentation	1358
12.81 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference	1358
12.81.1 Detailed Description	1358
12.81.2 Macro Definition Documentation	1359

12.82 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference	1359
12.82.1 Detailed Description	1365
12.82.2 Macro Definition Documentation	1365
12.82.3 Typedef Documentation	1367
12.82.4 Function Documentation	1367
12.83 cfe/modules/core_api/fsw/inc/cfe_es.h File Reference	1367
12.83.1 Detailed Description	1370
12.83.2 Macro Definition Documentation	1370
12.84 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference	1371
12.84.1 Detailed Description	1372
12.84.2 Macro Definition Documentation	1372
12.84.3 Typedef Documentation	1374
12.84.4 Enumeration Type Documentation	1375
12.85 cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference	1376
12.85.1 Detailed Description	1377
12.85.2 Macro Definition Documentation	1377
12.86 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference	1377
12.86.1 Detailed Description	1378
12.86.2 Macro Definition Documentation	1378
12.86.3 Typedef Documentation	1379
12.87 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference	1380
12.87.1 Detailed Description	1380
12.88 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference	1380
12.88.1 Detailed Description	1381
12.88.2 Typedef Documentation	1381
12.88.3 Enumeration Type Documentation	1382
12.89 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference	1383
12.89.1 Detailed Description	1385
12.90 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference	1385
12.90.1 Detailed Description	1386
12.90.2 Macro Definition Documentation	1386
12.90.3 Typedef Documentation	1387
12.90.4 Enumeration Type Documentation	1388
12.91 cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference	1389
12.91.1 Detailed Description	1390
12.91.2 Macro Definition Documentation	1390
12.91.3 Function Documentation	1391
12.92 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference	1396
12.92.1 Detailed Description	1396

12.92.2 Macro Definition Documentation	1396
12.93 cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference	1396
12.93.1 Detailed Description	1398
12.93.2 Macro Definition Documentation	1398
12.94 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference	1399
12.94.1 Detailed Description	1400
12.94.2 Macro Definition Documentation	1400
12.94.3 Typedef Documentation	1402
12.95 cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference	1402
12.95.1 Detailed Description	1403
12.96 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference	1403
12.96.1 Detailed Description	1405
12.96.2 Macro Definition Documentation	1405
12.96.3 Typedef Documentation	1405
12.96.4 Enumeration Type Documentation	1405
12.97 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h File Reference	1406
12.97.1 Detailed Description	1406
12.97.2 Macro Definition Documentation	1406
12.97.3 Typedef Documentation	1407
12.98 cfe/modules/core_api/fsw/inc/cfe_time.h File Reference	1407
12.98.1 Detailed Description	1408
12.98.2 Macro Definition Documentation	1409
12.99 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference	1409
12.99.1 Detailed Description	1409
12.99.2 Macro Definition Documentation	1409
12.99.3 Typedef Documentation	1410
12.99.4 Enumeration Type Documentation	1410
12.100 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference	1411
12.100.1 Detailed Description	1411
12.100.2 Macro Definition Documentation	1411
12.101 cfe/modules/es/config/default_cfe_es_extern_typedefs.h File Reference	1413
12.101.1 Detailed Description	1415
12.101.2 Macro Definition Documentation	1415
12.101.3 Typedef Documentation	1416
12.101.4 Enumeration Type Documentation	1419
12.102 cfe/modules/es/config/default_cfe_es_fcncodes.h File Reference	1421
12.102.1 Detailed Description	1422
12.102.2 Macro Definition Documentation	1422
12.103 cfe/modules/es/config/default_cfe_es_interface_cfg.h File Reference	1442

12.103.1 Detailed Description	1443
12.103.2 Macro Definition Documentation	1443
12.104 cfe/modules/es/config/default_cfe_es_internal_cfg.h File Reference	1445
12.104.1 Detailed Description	1447
12.104.2 Macro Definition Documentation	1447
12.105 cfe/modules/es/config/default_cfe_es_mission_cfg.h File Reference	1466
12.105.1 Detailed Description	1466
12.106 cfe/modules/es/config/default_cfe_es_msg.h File Reference	1466
12.106.1 Detailed Description	1466
12.107 cfe/modules/es/config/default_cfe_es_msgdefs.h File Reference	1466
12.107.1 Detailed Description	1468
12.107.2 Typedef Documentation	1468
12.107.3 Enumeration Type Documentation	1470
12.108 cfe/modules/es/config/default_cfe_es_msgids.h File Reference	1470
12.108.1 Detailed Description	1470
12.108.2 Macro Definition Documentation	1470
12.109 cfe/modules/es/config/default_cfe_es_msgstruct.h File Reference	1471
12.109.1 Detailed Description	1473
12.109.2 Typedef Documentation	1473
12.110 cfe/modules/es/config/default_cfe_es_platform_cfg.h File Reference	1475
12.110.1 Detailed Description	1475
12.111 cfe/modules/es/config/default_cfe_es_topicids.h File Reference	1476
12.111.1 Detailed Description	1476
12.111.2 Macro Definition Documentation	1476
12.112 cfe/modules/es/fsw/inc/cfe_es_eventids.h File Reference	1477
12.112.1 Detailed Description	1480
12.112.2 Macro Definition Documentation	1480
12.113 cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h File Reference	1502
12.113.1 Detailed Description	1502
12.113.2 Typedef Documentation	1502
12.113.3 Enumeration Type Documentation	1503
12.114 cfe/modules/evs/config/default_cfe_evs_fcncodes.h File Reference	1505
12.114.1 Detailed Description	1505
12.114.2 Macro Definition Documentation	1505
12.115 cfe/modules/evs/config/default_cfe_evs_interface_cfg.h File Reference	1523
12.115.1 Detailed Description	1523
12.115.2 Macro Definition Documentation	1523
12.116 cfe/modules/evs/config/default_cfe_evs_internal_cfg.h File Reference	1524
12.116.1 Detailed Description	1524

12.116.2 Macro Definition Documentation	1524
12.117 cfe/modules/evs/config/default_cfe_evs_mission_cfg.h File Reference	1528
12.117.1 Detailed Description	1528
12.118 cfe/modules/evs/config/default_cfe_evs_msg.h File Reference	1528
12.118.1 Detailed Description	1528
12.119 cfe/modules/evs/config/default_cfe_evs_msgdefs.h File Reference	1528
12.119.1 Detailed Description	1530
12.119.2 Macro Definition Documentation	1530
12.119.3 Typedef Documentation	1530
12.120 cfe/modules/evs/config/default_cfe_evs_msghids.h File Reference	1532
12.120.1 Detailed Description	1532
12.120.2 Macro Definition Documentation	1532
12.121 cfe/modules/evs/config/default_cfe_evs_msgstruct.h File Reference	1533
12.121.1 Detailed Description	1534
12.121.2 Typedef Documentation	1534
12.122 cfe/modules/evs/config/default_cfe_evs_platform_cfg.h File Reference	1536
12.122.1 Detailed Description	1536
12.123 cfe/modules/evs/config/default_cfe_evs_topicids.h File Reference	1536
12.123.1 Detailed Description	1536
12.123.2 Macro Definition Documentation	1536
12.124 cfe/modules/evs/fsw/inc/cfe_evs_eventids.h File Reference	1537
12.124.1 Detailed Description	1539
12.124.2 Macro Definition Documentation	1539
12.125 cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h File Reference	1549
12.125.1 Detailed Description	1549
12.126 cfe/modules/fs/config/default_cfe_fs_filedef.h File Reference	1549
12.126.1 Detailed Description	1550
12.126.2 Typedef Documentation	1550
12.126.3 Enumeration Type Documentation	1550
12.127 cfe/modules/fs/config/default_cfe_fs_interface_cfg.h File Reference	1551
12.127.1 Detailed Description	1551
12.127.2 Macro Definition Documentation	1551
12.128 cfe/modules/fs/config/default_cfe_fs_mission_cfg.h File Reference	1552
12.128.1 Detailed Description	1552
12.129 cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference	1552
12.129.1 Detailed Description	1552
12.129.2 Typedef Documentation	1552
12.130 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference	1553
12.130.1 Detailed Description	1553

12.131 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference	1553
12.131.1 Detailed Description	1554
12.131.2 Macro Definition Documentation	1554
12.132 cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h File Reference	1554
12.132.1 Detailed Description	1555
12.132.2 Macro Definition Documentation	1555
12.132.3 Typedef Documentation	1555
12.132.4 Enumeration Type Documentation	1556
12.133 cfe/modules/sb/config/default_cfe_sb_fcncodes.h File Reference	1556
12.133.1 Detailed Description	1557
12.133.2 Macro Definition Documentation	1557
12.134 cfe/modules/sb/config/default_cfe_sb_interface_cfg.h File Reference	1566
12.134.1 Detailed Description	1566
12.134.2 Macro Definition Documentation	1566
12.135 cfe/modules/sb/config/default_cfe_sb_internal_cfg.h File Reference	1567
12.135.1 Detailed Description	1568
12.135.2 Macro Definition Documentation	1568
12.136 cfe/modules/sb/config/default_cfe_sb_mission_cfg.h File Reference	1575
12.136.1 Detailed Description	1575
12.137 cfe/modules/sb/config/default_cfe_sb_msg.h File Reference	1575
12.137.1 Detailed Description	1576
12.138 cfe/modules/sb/config/default_cfe_sb_msgdefs.h File Reference	1576
12.138.1 Detailed Description	1577
12.138.2 Typedef Documentation	1577
12.139 cfe/modules/sb/config/default_cfe_sb_msgids.h File Reference	1578
12.139.1 Detailed Description	1579
12.139.2 Macro Definition Documentation	1579
12.140 cfe/modules/sb/config/default_cfe_sb_msgstruct.h File Reference	1580
12.140.1 Detailed Description	1580
12.140.2 Typedef Documentation	1580
12.141 cfe/modules/sb/config/default_cfe_sb_platform_cfg.h File Reference	1582
12.141.1 Detailed Description	1582
12.142 cfe/modules/sb/config/default_cfe_sb_topicids.h File Reference	1582
12.142.1 Detailed Description	1582
12.142.2 Macro Definition Documentation	1582
12.143 cfe/modules/sb/fsw/inc/cfe_sb_eventids.h File Reference	1583
12.143.1 Detailed Description	1586
12.143.2 Macro Definition Documentation	1586
12.144 cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h File Reference	1603

12.144.1 Detailed Description	1603
12.144.2 Typedef Documentation	1603
12.144.3 Enumeration Type Documentation	1603
12.145 cfe/modules/tbl/config/default_cfe_tbl_fcncodes.h File Reference	1604
12.145.1 Detailed Description	1604
12.145.2 Macro Definition Documentation	1604
12.146 cfe/modules/tbl/config/default_cfe_tbl_interface_cfg.h File Reference	1613
12.146.1 Detailed Description	1613
12.146.2 Macro Definition Documentation	1613
12.147 cfe/modules/tbl/config/default_cfe_tbl_internal_cfg.h File Reference	1614
12.147.1 Detailed Description	1615
12.147.2 Macro Definition Documentation	1615
12.148 cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h File Reference	1620
12.148.1 Detailed Description	1620
12.149 cfe/modules/tbl/config/default_cfe_tbl_msg.h File Reference	1620
12.149.1 Detailed Description	1620
12.150 cfe/modules/tbl/config/default_cfe_tbl_msghdefs.h File Reference	1621
12.150.1 Detailed Description	1622
12.150.2 Typedef Documentation	1622
12.151 cfe/modules/tbl/config/default_cfe_tbl_msghids.h File Reference	1623
12.151.1 Detailed Description	1623
12.151.2 Macro Definition Documentation	1623
12.152 cfe/modules/tbl/config/default_cfe_tbl_msghstruct.h File Reference	1624
12.152.1 Detailed Description	1625
12.152.2 Typedef Documentation	1625
12.153 cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h File Reference	1626
12.153.1 Detailed Description	1626
12.154 cfe/modules/tbl/config/default_cfe_tbl_topicids.h File Reference	1626
12.154.1 Detailed Description	1627
12.154.2 Macro Definition Documentation	1627
12.155 cfe/modules/tbl/inc/cfe_tbl_eventids.h File Reference	1627
12.155.1 Detailed Description	1630
12.155.2 Macro Definition Documentation	1630
12.156 cfe/modules/time/config/default_cfe_time_extern_typedefs.h File Reference	1647
12.156.1 Detailed Description	1648
12.156.2 Typedef Documentation	1649
12.156.3 Enumeration Type Documentation	1650
12.157 cfe/modules/time/config/default_cfe_time_fcncodes.h File Reference	1652
12.157.1 Detailed Description	1653

12.157.2 Macro Definition Documentation	1653
12.158 cfe/modules/time/config/default_cfe_time_interface_cfg.h File Reference	1668
12.158.1 Detailed Description	1668
12.158.2 Macro Definition Documentation	1668
12.159 cfe/modules/time/config/default_cfe_time_internal_cfg.h File Reference	1672
12.159.1 Detailed Description	1672
12.159.2 Macro Definition Documentation	1673
12.160 cfe/modules/time/config/default_cfe_time_mission_cfg.h File Reference	1677
12.160.1 Detailed Description	1678
12.161 cfe/modules/time/config/default_cfe_time_msg.h File Reference	1678
12.161.1 Detailed Description	1678
12.162 cfe/modules/time/config/default_cfe_time_msgdefs.h File Reference	1678
12.162.1 Detailed Description	1680
12.162.2 Typedef Documentation	1680
12.163 cfe/modules/time/config/default_cfe_time_msgids.h File Reference	1680
12.163.1 Detailed Description	1681
12.163.2 Macro Definition Documentation	1681
12.164 cfe/modules/time/config/default_cfe_time_msgstruct.h File Reference	1682
12.164.1 Detailed Description	1683
12.164.2 Typedef Documentation	1683
12.165 cfe/modules/time/config/default_cfe_time_platform_cfg.h File Reference	1685
12.165.1 Detailed Description	1685
12.166 cfe/modules/time/config/default_cfe_time_topicids.h File Reference	1685
12.166.1 Detailed Description	1685
12.166.2 Macro Definition Documentation	1686
12.167 cfe/modules/time/fsw/inc/cfe_time_eventids.h File Reference	1687
12.167.1 Detailed Description	1688
12.167.2 Macro Definition Documentation	1688
12.168 osal/docs/src/osal_frontpage.dox File Reference	1697
12.169 osal/docs/src/osal_fs.dox File Reference	1697
12.170 osal/docs/src/osal_timer.dox File Reference	1697
12.171 osal/src/os/inc/common_types.h File Reference	1697
12.171.1 Detailed Description	1698
12.171.2 Macro Definition Documentation	1699
12.171.3 Typedef Documentation	1699
12.171.4 Function Documentation	1701
12.172 osal/src/os/inc/osapi-binsem.h File Reference	1702
12.172.1 Detailed Description	1703
12.173 osal/src/os/inc/osapi-bsp.h File Reference	1703

12.173.1 Detailed Description	1703
12.174 osal/src/os/inc/osapi-clock.h File Reference	1703
12.174.1 Detailed Description	1705
12.174.2 Macro Definition Documentation	1705
12.174.3 Enumeration Type Documentation	1705
12.175 osal/src/os/inc/osapi-common.h File Reference	1706
12.175.1 Detailed Description	1707
12.175.2 Typedef Documentation	1707
12.175.3 Enumeration Type Documentation	1707
12.176 osal/src/os/inc/osapi-condvar.h File Reference	1708
12.176.1 Detailed Description	1709
12.177 osal/src/os/inc/osapi-constants.h File Reference	1709
12.177.1 Detailed Description	1709
12.177.2 Macro Definition Documentation	1709
12.178 osal/src/os/inc/osapi-countsem.h File Reference	1710
12.178.1 Detailed Description	1710
12.179 osal/src/os/inc/osapi-dir.h File Reference	1710
12.179.1 Detailed Description	1711
12.179.2 Macro Definition Documentation	1711
12.180 osal/src/os/inc/osapi-error.h File Reference	1711
12.180.1 Detailed Description	1713
12.180.2 Macro Definition Documentation	1713
12.180.3 Typedef Documentation	1714
12.181 osal/src/os/inc/osapi-file.h File Reference	1714
12.181.1 Detailed Description	1716
12.181.2 Macro Definition Documentation	1716
12.181.3 Enumeration Type Documentation	1717
12.182 osal/src/os/inc/osapi-fs.h File Reference	1717
12.182.1 Detailed Description	1718
12.182.2 Macro Definition Documentation	1718
12.183 osal/src/os/inc/osapi-heap.h File Reference	1719
12.183.1 Detailed Description	1719
12.184 osal/src/os/inc/osapi-idmap.h File Reference	1719
12.184.1 Detailed Description	1720
12.184.2 Macro Definition Documentation	1720
12.185 osal/src/os/inc/osapi-macros.h File Reference	1721
12.185.1 Detailed Description	1721
12.185.2 Macro Definition Documentation	1721
12.186 osal/src/os/inc/osapi-module.h File Reference	1722

12.186.1 Detailed Description	1723
12.186.2 Macro Definition Documentation	1723
12.187 osal/src/os/inc/osapi-mutex.h File Reference	1724
12.187.1 Detailed Description	1724
12.188 osal/src/os/inc/osapi-network.h File Reference	1724
12.188.1 Detailed Description	1725
12.189 osal/src/os/inc/osapi-printf.h File Reference	1725
12.189.1 Detailed Description	1725
12.190 osal/src/os/inc/osapi-queue.h File Reference	1725
12.190.1 Detailed Description	1726
12.191 osal/src/os/inc/osapi-select.h File Reference	1726
12.191.1 Detailed Description	1726
12.191.2 Enumeration Type Documentation	1727
12.192 osal/src/os/inc/osapi-shell.h File Reference	1727
12.192.1 Detailed Description	1727
12.193 osal/src/os/inc/osapi-sockets.h File Reference	1727
12.193.1 Detailed Description	1729
12.193.2 Macro Definition Documentation	1729
12.193.3 Enumeration Type Documentation	1729
12.194 osal/src/os/inc/osapi-task.h File Reference	1730
12.194.1 Detailed Description	1731
12.194.2 Macro Definition Documentation	1731
12.194.3 Typedef Documentation	1732
12.194.4 Function Documentation	1732
12.195 osal/src/os/inc/osapi-timebase.h File Reference	1732
12.195.1 Detailed Description	1733
12.195.2 Typedef Documentation	1733
12.196 osal/src/os/inc/osapi-timer.h File Reference	1733
12.196.1 Detailed Description	1734
12.196.2 Typedef Documentation	1734
12.197 osal/src/os/inc/osapi-version.h File Reference	1734
12.197.1 Detailed Description	1735
12.197.2 Macro Definition Documentation	1735
12.197.3 Function Documentation	1736
12.198 osal/src/os/inc/osapi.h File Reference	1737
12.198.1 Detailed Description	1738
12.199 psp/fsw/inc/cfe_psp.h File Reference	1738
12.199.1 Function Documentation	1739
12.200 psp/fsw/inc/cfe_psp_cache_api.h File Reference	1739

12.200.1 Function Documentation	1739
12.201 psp/fsw/inc/cfe_psp_cds_api.h File Reference	1739
12.201.1 Function Documentation	1740
12.202 psp/fsw/inc/cfe_psp_eepromaccess_api.h File Reference	1741
12.202.1 Function Documentation	1741
12.203 psp/fsw/inc/cfe_psp_error.h File Reference	1744
12.203.1 Detailed Description	1744
12.203.2 Macro Definition Documentation	1745
12.203.3 Typedef Documentation	1746
12.203.4 Function Documentation	1746
12.204 psp/fsw/inc/cfe_psp_exception_api.h File Reference	1747
12.204.1 Function Documentation	1747
12.205 psp/fsw/inc/cfe_psp_id_api.h File Reference	1749
12.205.1 Function Documentation	1749
12.206 psp/fsw/inc/cfe_psp_memaccess_api.h File Reference	1749
12.206.1 Function Documentation	1750
12.207 psp/fsw/inc/cfe_psp_memrange_api.h File Reference	1753
12.207.1 Macro Definition Documentation	1754
12.207.2 Function Documentation	1755
12.208 psp/fsw/inc/cfe_psp_port_api.h File Reference	1759
12.208.1 Function Documentation	1759
12.209 psp/fsw/inc/cfe_psp_ssr_api.h File Reference	1761
12.209.1 Function Documentation	1762
12.210 psp/fsw/inc/cfe_psp_timertick_api.h File Reference	1762
12.210.1 Macro Definition Documentation	1763
12.210.2 Function Documentation	1763
12.211 psp/fsw/inc/cfe_psp_version_api.h File Reference	1764
12.211.1 Function Documentation	1765
12.212 psp/fsw/inc/cfe_psp_watchdog_api.h File Reference	1765
12.212.1 Macro Definition Documentation	1767
12.212.2 Function Documentation	1769
Index	1771

1 CFDP (CF) Documentation

- CFS CFDP Introduction
- CFS CFDP Overview

- [CFS CFDP Design](#)
- [CFS CFDP Operation](#)
- [CFS CFDP Deployment Guide](#)
- [CFS CFDP Configuration](#)
- [CFS CFDP Table Definitions](#)
- [CFS CFDP Commands](#)
- [CFS CFDP Telemetry](#)
- [CFS CFDP Events](#)
- [CFS CFDP Constraints](#)
- [CFS CFDP Frequently Asked Questions](#)
- [CFS CFDP Lessons Learned](#)

1.1 CFS CFDP Introduction

Scope

This document provides a design and operational overview along with a complete specification for the commands and telemetry associated with the CFS CFDP (CF) application software.

The document is intended primarily for users of the software (operations personnel, test engineers, and maintenance personnel). The deployment guide section, is intended for mission developers when deploying and configuring the CF application software for a mission flight software build environment.

[CFS CFDP Version](#)

Acronyms

Acronym	Description
API	Application Programming Interface
ATP	Absolute Time Processor
ATS	Absolute Time tagged command Sequence
CCSDS	Consultative Committee for Space Data Systems
C&DH	Command and Data Handling
CFE	Core Flight Executive
CFS	Core Flight System
CI	Command Ingest
Cmd	Command
CPU	Central Processing Unit
EDAC	Error Detection and Correction
FDS	Flight Data System
FM	File Manager

FSW	Flight Software
GN&C	Guidance Navigation & Control
GSFC	Goddard Space Flight Center
HK	Housekeeping
HW, H/W	Hardware
ICD	Interface Control Document
ISR	Interrupt Service Routine
OS	Operating System
OSAL	Operating System Abstraction Layer
Pkts	Packets
RAM	Random-Access Memory
RTOS	Real Time Operating System
RTP	Relative Time Processor
RTS	Relative Time tagged command Sequence
SB	Software Bus Service
SBC	Single Board Computer
SC	Stored Commands task
SW, S/W	Software
TBD	To Be Determined
TBL	Table
TLM	Telemetry
UTC	Universal time code

Prev: [CFDP \(CF\) Documentation](#)

Next: [CFS CFDP Overview](#)

1.2 CFS CFDP Overview

CF is a cFS application for providing CFDP (CCSDS File Delivery Protocol) services that was designed to interface to the Core Flight Executive (cFE). Its primary function is to provide file receive and transmit functionality to this protocol. It works by mapping CFDP PDUs on and off cFS's software bus.

CF is a highly configurable application that is designed to be used on a wide range of flight missions. CF obtains its initial configuration through a configuration table and its platform and mission configuration files. The table contains default configuration settings and is loaded during CF initialization. The platform and mission configuration files are compile-time configuration parameters.

To transfer files using CFDP, the CF application must communicate with a CFDP compliant peer. CF may be configured to have any number of peers. The ASIST and ITOS ground systems contain a compliant peer that may be used for flight to ground (and ground to flight) transfers.

CF sends and receives file information and file-data in Protocol Data Units (PDUs) that are compliant with the CFDP standard protocol defined in the CCSDS 727.0-B-5 Blue Book. The PDUs are transferred to and from the CF application via CCSDS packets on the software bus. The system must be configured to get the PDU packets from the peer to the software bus (and vice-versa).

On a typical spacecraft using the cFE, science files and engineering files are continuously being created and queued for downlink. When transmission begins, the files are converted into a series of PDUs by CF, one after another essentially creating a continuous stream of file-data PDUs.

The CFS CFDP application is a CFDP implementation written specifically for CFS. The goal was bounded, small, time and space-efficient implementation of the CFDP features necessary for flight. CF implements Class 1 and 2 send and receive, as well as logic to cancel, suspend, resume, abandon, freeze, etc. Messages utilize "zero-copy" software bus buffers. CF configuration is mainly through the configuration table.

There are special features listed in the CFDP standard that are not applicable to flight software and are therefore not supported by this version of CF. See the constraints section of this document for more information.

Prev: [CFS CFDP Introduction](#)
Next: [CFS CFDP Design](#)

1.3 CFS CFDP Design

The CF Application has an operational interface consisting of one table, two different telemetry messages and twenty commands.

CF is an event-driven, single-threaded application that wakes up when one of the following four messages are received on its software bus pipe. Ground command, Housekeeping Request command, Incoming PDU or the Wake-up command. The Wake-up command tells CF to do file transaction processing. This command is typically sent periodically by the scheduler. The amount of file-transaction processing that is executed when this command is received, is configurable through the table parameter engine-cycles per wake-up.

Key differences from CF v2.X

- There's no more text-based ground commands where strings are sent for commands.
- There's no more memory pool. Everything is set up with limits both in platform config for static memory limits and the configuration table for dynamic timing and functionality limits.
- Much smaller code footprint. CF 3.0 is light-weight flight-only app. It does not provide any ground engine support.

For simplicity, the examples throughout this document refer to a typical operational scenario whereby the peer to the CF application is located on the ground. The CF application knows only of incoming file transactions and outgoing file transactions. The terms uplink, downlink and playback are often used, but only apply when the peer is located on the ground. CF may be configured to have more than one peer. One peer may be located onboard the spacecraft while another is located on the ground.

Prev: [CFS CFDP Overview](#)
Next: [CFS CFDP Operation](#)

1.4 CFS CFDP Operation

Initialization

CF initialization is the same for Power On Resets and Processor Resets. Standard application initialization activities are performed such as status and tracking initialization, message initialization and subscription, table initialization and registering with event services.

Outgoing Messages

The priority of CF's use of available outgoing messages on each wakeup is:

1. Send any pending message required for RX transaction
2. Re-send any message required for TX transaction if timer expired (for example, send EOF, or re-send EOF if ACK timer expired)
3. Send a filedata PDU in response to a NAK request. Per wakeup, one will be sent per each transaction in the TX wait state.
4. Once all TX wait transactions have processed their needs to send, the currently active TX transfer will send new filedata PDUs. It will keep sending them until there are no more messages on that wakeup (or the throttling semaphore stops it.)

Incoming Messages

Operationally, the flow of input packets from ground into CF should be throttled at some rate appropriate for ingest. CF has a per-channel configuration item for max number of RX messages processed per wakeup, and both CF and the ingest app need to be able to handle this.

Temporary file and directory use

Currently the temporary directory is only used to store class 2 RX file data in temporary files on a transaction that has not yet received a metadata PDU. When/if the metadata is received for that transaction, OS_mv is used to transition the temporary file to the desired destination location. Note that OS_mv attempts a rename first (faster but does not work across file systems), and if that fails then attempts a copy/delete (slower). Due to this the most efficient temporary directory configuration is for it to be on the same file system as the destination. Note that currently this only impacts class 2 RX with an out of order metadata packet, but there's future work being considered to use the temporary location for all receipts with an atomic transfer of the file after successful reception and verification (where applicable) of all the file data.

Engine

The CF application has a single internal core referred to as the engine. The engine is capable of transmitting and receiving a configurable number of transactions simultaneously. The engine builds outgoing Protocol Data Units (PDUs) and interprets the incoming PDUs. It handles all details regarding the CFDP standard protocol which is defined in the CCSDS 727.0-B-5 Blue Book. The engine processes the file transactions when it is 'cycled'. The number of engine cycles per wake up is a configuration parameter defined in the table. At most one PDU will be sent on a single engine cycle. Typically, the peer node also has an engine. When CF is transferring files to and from the ground, the peer is sometimes referred to as the ground engine. When faults and timeouts occur, it is important to indicate which engine detected the event.

Starting a Transaction

To transfer a file from the ground to the spacecraft, a 'put' request is given to the ground engine. There is no ground command telling CF to 'get' a file. The first indication to CF that an uplink transaction has started, is the receipt of the first PDU sent by the ground and received by CF.

To transfer a file from the spacecraft to the ground, a transmit file ground command is sent to CF. This ground command translates into a 'put' request to the flight engine.

The CFDP protocol does not support the concept of a 'get' request. The request to transfer a file is always made with a 'put' request at the source peer (i.e. where the file is located).

Transaction Class

All transfers are sent and received in one of two modes, class 1 or class 2. The CF application is capable of sending and receiving in class 1 and class 2. Class 1 transfers are similar to UDP in that they send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt. Class 2 transfers are analogous to TCP.

Queue Entries

The CF application keeps track of files in queue entries. There is one queue entry per transaction. The queue entries contain information such as filename and path, priority, class, channel etc. about each transaction. For incoming file transactions, the queue entry starts on the incoming active queue and is moved to the incoming history queue when the transaction is complete. For outgoing transactions, all queue entries start out on the pending queue (in response to a playback file command for example). The queue entry is then moved to the outgoing active queue when the transaction begins and to the history queue when complete.

Queues

The CF application tracks pending transactions, active transactions and completed transactions in its queues. For downlink (or outgoing) transactions there are three queues per channel, a pending queue, an active queue and a history queue. All queues hold queue entries that are described in the 'Queue Entries' section of this document. When a request to downlink a file is received, a queue entry is created and placed on the pending queue. When the transaction begins, the corresponding queue entry is moved from the pending queue to the active queue. After the transaction completes, (whether successful or not) the queue entry is moved from the active queue to the history queue. The history queue has a fixed depth, defined by the user in the CF configuration table. If a transaction is added when the history queue is full, the oldest queue entry is deleted.

For uplink (or incoming) transactions there are two queues, an active queue and a history queue. Uplink transactions do not have a pending queue as with outgoing transactions. When an uplink transaction begins, a queue entry is added to the active queue. When the transaction completes, the queue entry is moved from the active queue to the history queue. The history queue depth is specified by the user in the CF configuration table. When the history queue is full and an active transaction finishes, the oldest entry on the history queue is deleted.

Incoming File Transactions

When files are transferred in the uplink direction, the ground peer receives the initial request to send the file. This action causes the ground peer to send a series of PDUs that are routed to the CF application. The CF application does not get a request to receive a file. The first indication to the CF application that an uplink transaction has started, is the receipt of the first PDU of the transaction.

The CF application is capable of receiving files in class 1 or class 2 mode on a per-file basis. The class mode is embedded in the PDUs received.

The message ID for incoming PDUs is defined in the CF configuration table.

When a file is uploaded to the spacecraft in class 2 mode, the CF app must acknowledge the receipt of the file by sending an acknowledgment PDU to the ground. This response must be sent on a specified output channel (output channels are described later). The channel number for this response is defined by the user in the configuration table.

The CF application keeps a list of all incoming transactions in its internal queues. There are two queues designated for incoming transfers. The incoming active queue holds information about all incoming transactions that are currently active. The incoming history queue holds information about all incoming transactions that are complete. The full contents of either queue can be viewed on command. The depth of the history queue is defined in the table.

Outgoing File Transactions

All outgoing file transactions are initiated by the CF application in response to a playback file command, a playback directory command or a file found in a polling directory. The peer entity does not request to receive a file. All outgoing file transactions are inserted into a pending queue by CF before they are actually sent. The CF application reads the pending queue (if reading is enabled) and starts the next transaction immediately after the data from the previous file has been sent. This process of queueing files and sending them sequentially, prevents the engine from being inundated when the user requests to send multiple files. Once the transaction begins, the queue entry is moved to the outgoing active queue and then to the outgoing history queue when it's complete. The engine processes the outgoing file transactions when it is 'cycled'. The number of engine cycles per wake up is defined in the table. At most one PDU will be sent on a single engine cycle.

Output Channels

The CF application supports sending files to a configurable number of destinations. The output channels are configured through table parameters. Each channel has a pending queue, active queue and history queue. All queue entries for outgoing transactions start out on the pending queue, then get moved to the active queue when the transaction begins. After the transaction is complete the queue entry is moved to the history queue. The queues may be viewed by command at any time. The pending queue reads may be enabled or disabled at any time. Each channel has a dedicated throttling semaphore, peer entity ID, message ID for outgoing PDUs and a configurable number of polling directories. File output transactions may occur simultaneously on different channels. The engine processes all active outgoing transactions in a round-robin fashion so as not to starve any one transaction. CF is not capable of prioritizing across channels.

Queueing Files for Output

There are three ways to request a file (or files) to be sent. The file transmit command, the directory playback command or through poll directory processing. The CF polling directory feature continually checks a directory for files and after detecting a new file in the directory, inserts a queue entry containing the file name (and other info) on the pending queue.

Priority

Each file-send transaction has an associated priority which is specified by the user. The priority of the transaction determines where it is inserted in the pending queue. High priority transactions get inserted toward the front of the queue. There are 256 levels of priority, zero being the highest. Priority is given as a command parameter for the playback file command and the playback directory command. For poll directory processing, each polling directory has an associated priority given as a table parameter. Please note that this priority applies only within a channel. CF does not support prioritization across channels. Prioritization across channels (if needed) would typically be implemented by the application receiving the PDUs.

Preserve Setting

When an outgoing file transaction is successfully complete, the user may want the file to be deleted by CF. The preserve setting allows the user to specify whether the file is deleted or not. The preserve setting gives two choices, delete or keep. This setting is specified as a parameter in the transmit file command, the playback directory command and on each polling directory. If a file transaction is not successful, the file cannot be deleted by CF.

Throttling Semaphore

Throttling outgoing PDUs may be necessary when the application that receives the outgoing PDUs (typically TO) needs to control the flow of packets. The throttling semaphore is a counting semaphore that is shared between another application and CF. Throttling may be configured as in-use or not-in-use on a per-channel basis. To configure as in-use, the receiving app must create a counting semaphore during initialization, using the name defined in the CF table. After creation, the receiving app must 'give' the semaphore each time it is ready to receive a PDU. On the CF side, CF attempts to get the semaphore ID by calling an OSAL function to Get-SemaphoreID-by-Name during CF initialization. The name defined in the table is given as a parameter to this call. CF has code to ensure that this call is executed after the receiving app initializes. If the attempt to Get-SemaphoreID-by-Name fails, then throttling on that channel is not-in-use and PDUs are sent whenever the engine has a PDU ready to output. If successful, each time the engine has a PDU to output, CF will attempt a non-blocking 'take' on the throttling semaphore. If the 'take' is successful, the green light counter in telemetry is incremented and the PDU is sent on the software bus. If the 'take' is not successful, the PDU is held by the engine, the red-light counter is incremented and the 'take' is called again on the next engine cycle.

Polling Directories

A polling directory is a directory that is polled by CF periodically. CF does not create these directories. They must be created before they can be enabled. When files are found in a polling directory that is enabled, the files are automatically queued by CF for output. The polling rate is configurable and each channel has a configurable number of polling directories. Each polling directory has an enable, a class setting, a priority setting, a destination directory. When enabled, CF will periodically check the polling directories for files. When a file is found, CF will place the file information on the pending queue if it is closed and not already on the queue and not currently active. All polling directories are checked at the same frequency which is defined by the table parameter NumWakeupsPerPollDirChk. Setting this parameter to one will cause CF to check the polling directories at the fastest possible rate, every time it receives a 'wake-up' command. Checking polling directories is a processor-intensive effort, it is best to keep the polling rate as low as possible.

Failed Transmit and directory use

Currently, the fail directory is used only to store class 2 TX files that failed during transmission from a polling directory. When a file fails during transmission from a polling directory, it is deleted from the polling directory to prevent an infinite loop and moved to the fail directory. However, if a file fails during a class 2 TX outside of a polling directory, the file will not be deleted.

Efficiency

The CF application can be a processor intensive application. Some operating systems have significant overhead associated with file system operations. Opening and closing a file, querying the file system for file size, deleting the file, opening directories, looping through a directory list can consume a considerable amount of processor time. For this reason, transmitting small files at a high rate for long periods of time may be a worst-case-timing scenario. File system overhead is less of an issue when file sizes are large. The terms 'large' and 'small' used here are relative to the downlink rate. With a downlink rate of 1 Mbps for example, a good file size would be 1 MByte or larger.

Also, it is best to keep the number of files on the pending queue to a minimum. When the number of files on the pending queue is high, (such as hundreds) prioritization and standard checking causes CF processing to be significant each time a file is added to the queue.

Polling directory processing is also subject to file system overhead. It is recommended that the rate of poll processing be kept low and unused polling directories should be disabled.

Memory Use

CF uses statically allocated memory based on configuration parameters defined in the platform configuration file. This static memory is used for queue entries. The life cycle of a queue entry begins when a request to queue a file for downlink is received. Or in the case of incoming transactions, the queue entry is reserved when the meta-data PDU is received by CF. For the incoming transaction case, the queue entry starts out on the incoming active queue, then the entry is moved to the history queue when the transaction completes. For outgoing transactions, the queue entry starts out on the pending queue, then moves to the active queue when the transaction begins, then moves to the history queue when the transaction is complete.

The history queue has a sliding window effect. When the queue is full and a new transaction needs to be added, the oldest transaction will be removed, making room for the new transaction.

The history queue depth is a configuration parameter and specified in the CF configuration table. When the queue entry is 'pushed-off' the history queue, the memory for the queue will be available.

For incoming file transactions, CF writes file data from the incoming packet directly to the file without the use of an internal buffer. Incoming transaction size is limited by the underlying transport layer.

For outgoing file transactions CF requests a buffer from SB sized for CF_MAX_PDU_SIZE, the CCSDS header size, and CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES. Outgoing packets are then filled up to this maximum value for transmission. Note the maximum outgoing size can also be limited by the table or set parameter command for outgoing_file_chunk_size.

Generation of packets can be flow controlled via a throttling semaphore defined in the CF configuration table.

Prev: [CFS CFDP Design](#)

Next: [CFS CFDP Deployment Guide](#)

1.5 CFS CFDP Deployment Guide

Follow the general guidelines below for platform deployment of the CFDP app.

Configuration

CF uses two sets of configuration parameters: compile-time configurable parameters in the cf_platform_cfg.h file and run-time configurable parameters in the [cf_def_config.c](#) file. Most parameters are included in the [cf_def_config.c](#) file for maximum flexibility.

cf_platform_cfg.h uses macro definitions for configurable options, while [cf_def_config.c](#) uses a table. [cf_def_config.c](#) parameters can be accessed using the get/set functions [CF_GetParamCmd\(\)](#) and [CF_SetParamCmd\(\)](#).

CF expects to receive a CF_WAKEUP_MID message from the SCH (scheduler) app at a fixed rate. The number of wakeups per second is reflected in the configuration table. This drives CF's timing.

Channels CF version 3.0 has a concept of "channels" which have their own configuration. The channel configuration is done in the [cf_def_config.c](#) file and allows message IDs for incoming and outgoing PDUs to be unique per channel. Each channel can be configured with polling directories as well.

Flow Control By default, CF assumes that a per-channel semaphore is provided by the Telemetry Output (TO) application or an equivalent application. The semaphore name is defined in the CF configuration table and must match the name of the semaphore created by TO. If TO does not create a semaphore, the semaphore name can be left as an empty string in the configuration table indicating that a semaphore should not be used. If a semaphore is expected and not found, the application will terminate during initialization. If no semaphore is used, the outgoing messages per wakeup may need to be more limited. It's a valid configuration to have both the semaphore and maximum outgoing messages per wakeup as well.

Integration

Software Bus Each channel has an input message ID that is subscribed on, and a message ID to publish its messages on.

Scheduler CF as a whole expects the CF_WAKEUP_MID message as described earlier at a fixed rate. If the number of wakeups per second is changed in SCH, then the ticks_per_second configuration parameter in the CF configuration table must also be updated.

Endianness CF is endian agnostic and no longer requires specific compile time configuration/defines to handle different endian systems.

Integration with TO TO's pipe needs to be able to receive packets of [CF_MAX_PDU_SIZE](#)

Prev: [CFS CFDP Operation](#)
Next: [CFS CFDP Configuration](#)

1.6 CFS CFDP Configuration

The CF application provides the user with a set of compile-time configuration parameters as well as a set of run-time configuration parameters. All compile-time configuration parameters are specified in headers files. There are three header files used by the CF application, cf_platform_cfg.h, [cf_perfids.h](#) and cf_msgids.h. For details regarding the compile-time configuration parameters, refer to the header files.

[CFS CFDP Mission Configuration](#)

[CFS CFDP Platform Configuration](#)

All run-time configuration parameters are specified in the CF configuration table.

[CFS CFDP Table Definitions](#)

Prev: [CFS CFDP Deployment Guide](#)

Next: [CFS CFDP Commands](#)

1.7 CFS CFDP Table Definitions

The CF application has one table used for configuration. This table is accessed during initialization and updated by CF when changes to the table parameters are made through CF commands or CFE table services.

The table contains default configuration settings. Many table configuration settings can be adjusted by command. These adjustments will modify the table and are therefore reflected if the table is dumped. These adjustments will also change the table checksum. The configuration table is loaded at the time the application is started. CF supports table updates during runtime only when the engine is disabled.

CF utilizes a CFS table for run-time configuration defined by [CF_ConfigTable_t](#). The channel configuration is in [CF_ConfigTable_t.chan](#) which contains a polling element defined by [CF_PollDir_t](#).

Prev: [CFS CFDP Configuration](#)

Next: [CFS CFDP Constraints](#)

1.8 CFS CFDP Commands

[CFS CFDP Command Descriptions](#)

[CFS CFDP Command Message IDs](#)

[CFS CFDP Command Structures](#)

[CFS CFDP Command Codes](#)

Prev: [CFS CFDP Table Definitions](#)

Next: [CFS CFDP Telemetry](#)

1.8.1 CFS CFDP Command Descriptions

Noop Command

The CF Noop command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_NOOP_CC](#). This command is useful for verifying that the command interface to the CF Application is working. It also causes an event message to be generated that contains the CF Application's version information. It should be noted that the version information can also be obtained when the Application starts up. After CF has successfully initialized itself, an event message is generated that indicates successful initialization and also includes the Application's version information. Both of these event messages are 'Informational' and are NOT filtered by default.

Reset Counters Command

The CF Reset counters command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_RESET_CC](#). This command is used to reset the counters in the housekeeping telemetry packet. All counters can be reset or they can be reset by category. The categories are command counters, fault counters, incoming file counters and outgoing file counters.

When the command is executed successfully, the command counter will be zero and an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
* \par Command Structure
*   #CF_UnionArgs_Payload_t where byte[0] specifies the counters type, byte[1-3] don't care:
*   - 0 = all counters
*   - 1 = command counters
*   - 2 = fault counters
*   - 3 = up counters
*   - 4 = down counters
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
}
CF_UnionArgs_Payload_t;
```

The command parameter byte[0] identifies which category of counters to reset.counters. The value should be set to one of five possible values defined in the reset enumeration.

```
typedef enum {
    CF_Reset_all      = 0,
    CF_Reset_command = 1,
    CF_Reset_fault   = 2,
    CF_Reset_up       = 3,
    CF_Reset_down     = 4
} CF_Reset_t;
```

Transmit File Command

The CF Transmit File command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_TX_FILE_CC](#).

This command is used to queue a file to be sent by the CF application. To 'transmit' a file means to output, or send a file.

When the command is executed successfully, the command counter will increment and an event will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure. File will not be deleted if the 'transmit' failed (unless in an polling directories).

```
typedef struct CF_TxFileCmd
{
    CFE_MSG_CommandHeader_t cmd_header;
    uint8                  cfdp_class;
    uint8                  keep;
    uint8                  chan_num;
    uint8                  priority;
    CF_EntityId_t          dest_id;
    char                   src_filename[CF_FILENAME_MAX_LEN];
    char                   dst_filename[CF_FILENAME_MAX_LEN];
} CF_TxFileCmd_t;
```

The first parameter, `cfdp_class`, identifies whether the file will be transferred using [CF_CFDP_CLASS_1](#) or [CF_CFDP_CLASS_2](#). For Class 1 transfers, CF will send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt.

The second parameter, `keep`, specifies whether the file will be deleted or preserved by CF after the transfer successfully completes. To have the file deleted by CF, set this parameter to zero. If this parameter is set to one, the file will not be deleted. Regardless of the setting, if the file-transfer is not successful the file will not be deleted.

The third parameter, `chan_num`, specifies the output channel in which the file will be sent. The value range for this parameter is 0 to ([CF_NUM_CHANNELS](#) - 1). [CF_NUM_CHANNELS](#) is specified in the CF platform configuration file.

The fourth parameter, `priority`, specifies where the file is placed on the pending queue. High priority files are placed at the front of the queue. A value of zero is the highest priority. A value of 255 is the lowest priority.

The fifth parameter, `src_filename`, specifies the path name and filename to send. This parameter is a string with max size equal to [CF_FILENAME_MAX_LEN](#) bytes. The `src_filename` must be an existing file. The string must begin with a forward slash, have no spaces and be properly terminated.

The last parameter, `dst_filename`, specifies the destination path name and filename. This parameter is a string with max size equal to [CF_FILENAME_MAX_LEN](#) bytes. This parameter is delivered to the peer so that the peer knows where to store the file. The peer engine dictates the requirements of this string. The CF application allows this string to be NULL in which case the peer engine will store the incoming file in the default directory. If the string is not NULL, CF requires that it is properly terminated and contains no spaces. This parameter can be used to rename the file after it's received at the destination.

Playback Directory Command

The CF Playback Directory command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_PLAYBACK_DIR_CC](#). The command causes an event message to be generated. This command is used to queue the files that are located in the specified directory at the time the command is received.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

To 'playback' a directory means to send all files in that directory. To queue the files for sending, the class must be 1 or 2, the channel must be in-use, the files must be closed and not be active or pending, the preserve parameter must be 0 (Delete) or 1 (Keep), the path names must include no spaces and be properly terminated. The src_filename must begin and end with a forward slash. All possible values for Priority are valid. A priority value of zero is the highest priority.

```
typedef CF_TxFileCmd_t CF_PlaybackDirCmd_t;
typedef struct CF_TxFileCmd
{
    CFE_MSG_CommandHeader_t cmd_header;
    uint8                  cfdp_class;
    uint8                  keep;
    uint8                  chan_num;
    uint8                  priority;
    CF_EntityId_t          dest_id;
    char                   src_filename[CF_FILENAME_MAX_LEN];
    char                   dst_filename[CF_FILENAME_MAX_LEN];
} CF_TxFileCmd_t;
```

The first parameter, `cfdp_class`, identifies whether the files will be transferred using [CF_CFDP_CLASS_1](#) or [CF_CFDP_CLASS_2](#). For Class 1 transfers, CF will send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt.

The second parameter, `keep`, specifies whether the files will be deleted or preserved by CF after the transfer successfully completes. To have the files deleted by CF, set this parameter to zero. If this parameter is set to one, the file will not be deleted. Regardless of the setting, if the file-transfer is not successful the file will not be deleted.

The third parameter, `chan_num`, specifies the output channel in which the files will be sent. All files in the directory will be sent on the specified channel. The value range for this parameter is 0 to ([CF_NUM_CHANNELS](#)

- 1). [CF_NUM_CHANNELS](#) is specified in the CF platform configuration file.

The fourth parameter, `priority`, specifies where the files are placed on the pending queue. All files in the directory will be queued with the given priority. High priority files are placed at the front of the queue. A value of zero is the highest priority. A value of 255 is the lowest priority.

The fifth parameter, `src_filename`, specifies the path name where the files are located. The string must have no spaces, be properly terminated and have a forward slash as the last character. This parameter is a string with max size equal to [CF_FILENAME_MAX_LEN](#) characters.

The last parameter, `dst_filename`, specifies where the files are to be stored after they are received by the peer. This parameter is a string with max size equal to [CF_FILENAME_MAX_LEN](#) bytes. This parameter is delivered to the peer so that the peer knows where to store the file. The peer engine dictates the requirements of this string. The CF application allows this string to be NULL in which case the peer engine will store the incoming files in the default directory. If the string is not NULL, CF requires that it is properly terminated, contains no spaces and ends with a forward slash. There is no way to rename the files at the destination as in the Playback File command.

Freeze Command

The CF Freeze command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_FREEZE_CC](#). The freeze command has no command parameters. This command is used to freeze all transactions. The freeze command should be applied to both the source and destination peers at nearly the same time.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

Thaw Command

The CF Thaw command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_THAW_CC](#). The thaw command has no command parameters. This command is used to thaw all transactions that were commanded to 'freeze' earlier. The thaw command should be applied to both the source and destination peers at nearly the same time.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

Suspend Command

The CF Suspend command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_SUSPEND_CC](#). This command is used to suspend one or all transactions. The suspend command has one command parameter that indicates what transaction to suspend. See details below. The suspend command should be applied to both the source and destination peers at nearly the same time.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

NOTE: Suspending an outgoing transaction before EOF is sent, will pause the flow of PDUs on that channel. This happens because the next file is started when the current file EOF is sent. If a user wishes to stop the current transaction (before the EOF is sent) and still allow the next pending file to begin, the current transaction should be cancelled (or abandoned) in lieu of being suspended. Canceling is always a better option than abandoning.

NOTE: When a suspended transaction is cancelled, the cancel does not take effect until the transaction is resumed.

```
typedef struct CF_Transaction_Payload
{
    CF_TransactionSeq_t      ts;
    CF_EntityId_t           eid;
    uint8                   chan;
    uint8                   spare[3];
} CF_Transaction_Payload_t;
```

The `ts` parameter specifies the transaction sequence number to suspend. The `eid` parameter specifies the entity id used in the transaction. The `chan` parameter can specify a single channel, all channels, or the channel corresponding to the transaction sequence number.

Resume Command

The CF Resume command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_RESUME_CC](#). This command is used to resume a suspended transaction or all transactions. The resume command has one command parameter that indicates what transaction to resume. See details below. The resume command should be applied to both the source and destination peers at nearly the same time.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
typedef struct CF_Transaction_Payload
{
    CF_TransactionSeq_t      ts;
    CF_EntityId_t           eid;
    uint8                   chan;
    uint8                   spare[3];
} CF_Transaction_Payload_t;
```

The `ts` parameter specifies the transaction sequence number to resume. The `eid` parameter specifies the entity id used in the transaction. The `chan` parameter can specify a single channel, all channels, or the channel corresponding to the transaction sequence number.

Cancel Command

The CF Cancel command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_CANCEL_CC](#). This command is used to cancel a transaction or all transactions. The cancel command has one command parameter that indicates what transaction to cancel. See details below. The cancel command should be sent to the source entity only. For example, uplink transactions should be cancelled at the ground engine. The CF application should not receive a cancel command in this case. The CF application will learn of the cancel request through the protocol messages. Downlink transactions and outgoing transactions (with respect to CF) should be cancelled by sending this CF cancel command.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

NOTE: If a Cancel command is received by CF on an outgoing transaction that is suspended, the cancel does not take effect until the transaction is resumed.

```
typedef struct CF_Transaction_Payload
{
    CF_TransactionSeq_t      ts;
    CF_EntityId_t           eid;
    uint8                   chan;
    uint8                   spare[3];
} CF_Transaction_Payload_t;
```

The `ts` parameter specifies the transaction sequence number to cancel. The `eid` parameter specifies the entity id used in the transaction. The `chan` parameter can specify a single channel, all channels, or the channel corresponding to the transaction sequence number.

Abandon Command

The CF Abandon command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_ABANDON_CC](#). This command is used to abandon a transaction or all transactions. The abandon command has one command parameter that indicates what transaction to abandon. See details below. The abandon command should be applied to both the source and destination peers at nearly the same time.

When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure. When the command is executed successfully, the command counter is incremented and an event message will be generated. This event message is an 'Informational' type and is NOT filtered by default. If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

NOTE: Unlike the cancel command, if a suspended transaction is abandoned, the transaction will be abandoned at the time the abandon command is received. Likewise, if a frozen transaction is abandoned, the transaction will be abandoned when the abandoned cmd is received.

```
typedef struct CF_Transaction_Payload
{
    CF_TransactionSeq_t      ts;
    CF_EntityId_t           eid;
    uint8                   chan;
    uint8                   spare[3];
} CF_Transaction_Payload_t;
```

The `ts` parameter specifies the transaction sequence number to abandon. The `eid` parameter specifies the entity id used in the transaction. The `chan` parameter can specify a single channel, all channels, or the channel corresponding to the transaction sequence number.

Set MIB Parameter Command

The CF Set MIB Parameter command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_SET_PARAM_CC](#). This command is used to change the flight engine Message Information Base (MIB). The MIB is a term used in the CCSDS blue book that can be interpreted as the engine configuration parameters. The command has two command parameters, Param indicates which parameter to change, and Value indicates the new setting.

When the command is executed successfully, the command counter is incremented and an event message will be generated displaying the parameter values received. This event message is an 'Informational' type and is NOT filtered by default.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

This command may be used to change any flight MIB parameter.

NOTE: Changing these parameters will change the actual table values, thereby changing the checksum of the CF configuration table.

```
typedef struct CF_SetParamCmd
{
    CFE_MSG_CommandHeader_t cmd_header;
    uint32                  value;
    uint8                   key;
    uint8                   chan_num;
    uint8                   spare[2];
} CF_SetParamCmd_t;
```

The `value` parameter specifies the new value of the engine parameter.

The `key` parameter specifies which engine parameter to set.

The `chan_num` parameter specifies the channel number to set.

Get MIB Parameter Command

The CF Set MIB Parameter command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_GET_PARAM_CC](#). This command is used to view a single Message Information Base (MIB) parameter. The MIB is a term used in the CCSDS blue book that can be interpreted as the engine configuration parameters.

The command has two command parameter, `key` indicates which parameter to view and the `channel number` specifies the channel configuration to use. The parameter given and its current setting will be displayed in the event.

When the command is executed successfully, the command counter is incremented and an event message will be generated displaying the given parameter value and the current setting for that parameter. This event message is an 'Informational' type and is NOT filtered by default.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

This command may be used to view any flight MIB parameter.

```
typedef struct CF_GetParamCmd
{
    CFE_MSG_CommandHeader_t cmd_header;
    uint8                  key;
    uint8                  chan_num;
} CF_GetParamCmd_t;
```

The `key` parameter specifies which engine parameter to get.

The `chan_num` parameter specifies the channel number to get the data from.

Write Queue Information Command

The CF Write Queue Information command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_WRITE_QUEUE_CC](#).

This command is used to write the contents of a single queue to a file. CF has a pending, queue, an active queue and a history queue for each output channel. CF also has an active queue and a history queue for incoming transactions.

When the command is executed successfully, the command counter is incremented and an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
typedef struct CF_WriteQueueCmd
{
    CFE_MSG_CommandHeader_t cmd_header;
    uint8                  type;
    uint8                  chan;
    uint8                  queue;
    uint8                  spare;

    char filename[CF_FILENAME_MAX_LEN];
} CF_WriteQueueCmd_t;
```

The first parameter, `type`, specifies the queue type. The queue type may be uplink (incoming), or downlink (outgoing), or both,

The second parameter, `chan`, is necessary only if the type parameter is set to a value of two (downlink). If the Type parameter is set to a value of one (uplink), the code does not read this parameter. Chan specifies the downlink channel that owns the queue. The value range for this parameter is 0 to ([CF_NUM_CHANNELS](#) - 1). [CF_NUM_CHANNELS](#) is specified in the CF platform configuration file.

The third parameter, `queue`, identifies which queue contents will be written to the file. A value of 0 for pending queue, 1 for active and 2 for history. Because there is no uplink pending queue, a value of zero is not valid when the Type parameter is set to one (uplink).

The fourth parameter, `filename`, specifies the name of the file that will receive the queue data. This parameter is a string with max size equal to [CF_FILENAME_MAX_LEN](#) bytes specified in the CF platform configuration file.

Enable Dequeue Command

The CF Enable Dequeue command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_ENABLE_DEQUEUE_CC](#).

This command is used to enable reading from the pending queue on a particular channel. It has one parameter (channel) and is sent when the pending queue reads are disabled for that channel. The pending queue holds the names of the files that are waiting to be sent out by CF. This command has no effect on incoming file transactions.

When the command is executed successfully, the command counter is incremented and an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
*      Command Structure
*      #CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels
*      - 255 = all channels
*      - else = single channel
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
} CF_UnionArgs_Payload_t;
```

The first and only parameter, `byte[0]`, specifies which pending queue to enable. Each channel has one pending queue. The value range for this parameter is 0 to

1. A value of 255 specifies all channels. All other values are for a single channel.

Disable Dequeue Command

The CF Enable Dequeue command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_DISABLE_DEQUEUE_CC](#).

This command is used to disable reading from the pending queue on a particular channel. It has one parameter (channel) and sent when the user would like to stop sending files on a particular channel.

NOTE: This command does not stop a file transaction that is in progress on the specified channel. Use the cancel command to stop a file transaction that is in progress.

When the command is executed successfully, the command counter is incremented and then an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
*      Command Structure
*      #CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels
*      - 255 = all channels
*      - else = single channel
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
} CF_UnionArgs_Payload_t;
```

The first and only parameter, `byte[0]`, specifies which pending queue to enable. Each channel has one pending queue. The value range for this parameter is 0 to

1. A value of 255 specifies all channels. All other values are for a single channel.

Enable Directory Polling Command

The CF Enable Directory Polling command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_ENABLE_DIR_POLLING_CC](#).

This command is used to enable one or all polling directories on a particular channel. CF will check polling directories for filenames and queue them for output, only if the polling directory is enabled. Polling directory processing consumes a fair amount of CPU utilization. Opening directories periodically and looping through hidden files, closed files and files that have already been queued can take a substantial amount of CPU time. It is recommended to keep polling directories disabled when they are not actively receiving files.

When the command is executed successfully, the command counter is incremented and an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```

*      Command Structure
*      #CF_UnionArgs_Payload_t
*
*      byte[0] specifies the channel number or all channels
*      - 255 = all channels
*      - else = single channel
*
*      byte[1] specifies the polling directory index
*      - 255 = all polling directories
*      - else = single polling directory index
*
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
} CF_UnionArgs_Payload_t;
```

The first parameter, `byte[0]`, specifies which channel contains the polling directory to enable. The value range for this parameter is 0 to 255. A value of 255 specifies all channels. All other values are for a single channel.

The second parameter, `byte[1]`, specifies which polling directory to enable. To enable all polling directories on a specific channel, set this parameter to 0xFF. The polling directories are numbered from zero to (`CF_MAX_POLLING_DIRS_PER_CHAN` - 1). `CF_MAX_POLLING_DIRS_PER_CHAN` is specified in the CF platform configuration file. The polling directory number may be obtained by viewing the contents of the CF configuration table.

Disable Directory Polling Command

The CF Enable Directory Polling command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_DISABLE_DIR_POLLING_CC](#).

This command is used to disable one or all polling directories on a particular channel. CF will check polling directories for filenames and queue them for output, only if the polling directory is enabled. Polling directory processing consumes a fair amount of CPU utilization. Opening directories periodically and looping through hidden files, closed files and files that have already been queued can take a substantial amount of CPU time. It is recommended to keep polling directories disabled when they are not actively receiving files.

When the command is executed successfully, the command counter is incremented and an event message will be generated.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
*      Command Structure
*      #CF_UnionArgs_Payload_t
*
*      byte[0] specifies the channel number or all channels
*      - 255 = all channels
*      - else = single channel
*
*      byte[1] specifies the polling directory index
*      - 255 = all polling directories
*      - else = single polling directory index
*
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
} CF_UnionArgs_Payload_t;
```

The first parameter, `byte[0]`, specifies which channel contains the polling directory to disable. The value range for this parameter is 0 to 255. A value of 255 specifies all channels. All other values are for a single channel.

The second parameter, `byte[1]`, specifies which polling directory to disable. To disable all polling directories on a specific channel, set this parameter to 0xFF. The polling directories are numbered from zero to (`CF_MAX_POLLING_DIRS_PER_CHAN` - 1). `CF_MAX_POLLING_DIRS_PER_CHAN` is specified in the CF platform configuration file. The polling directory number may be obtained by viewing the contents of the CF configuration table.

Purge Queue Command

The CF Write Queue Information command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_PURGE_QUEUE_CC](#).

This command is used to remove all entries on a single queue. CF has a pending queue, an active queue, and a history queue for each channel. Only the pending and history queues can be purged.

When the command is executed successfully, the command counter is incremented and an event message will be generated indicating that the node has been removed.

If the command is not successful, the command error counter will increment and an error event will be generated indicating the reason for failure.

```
*      Command Structure
*      #CF_UnionArgs_Payload_t
*
*      byte[0] specifies the channel number or all channels
*      - 255 = all channels
*      - else = single channel
*
*      byte[1] specifies the queue
*      - 0 = Pending queue
*      - 1 = History queue
*      - 2 = Both pending and history queue
*
typedef union CF_UnionArgs_Payload
{
    uint32 dword;
    uint16 hword[2];
    uint8 byte[4];
} CF_UnionArgs_Payload_t;
```

The first parameter, `byte[0]`, specifies a single channel id or a value indicating all channels.

The second parameter, `byte[1]`, identifies which queue will be purged. A value of 0 for pending queue, 1 for history and 2 for both.

Enable Engine Command

The CF Enable Engine command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_ENABLE_ENGINE_CC](#). The command has no command parameters and is used to reinitialize engine and enable processing. Note configuration table updates are not processed while the engine is enabled.

Enable Disable Command

The CF Enable Engine command is sent to CF using message ID [CF_CMD_MID](#) with command code [CF_DISABLE_ENGINE_CC](#). The command has no command parameters and is used to disable engine processing. Note configuration table updates can be performed while the engine is disabled, and when the engine is re-enabled the new configuration will take effect.

Prev: [CFS CFDP Configuration](#)

Next: [CFS CFDP Telemetry](#)

1.9 CFS CFDP Telemetry

[CFS CFDP Telemetry Descriptions](#)

[CFS CFDP Telemetry Message IDs](#)

[CFS CFDP Telemetry](#)

Prev: [CFS CFDP Commands](#)

Next: [CFS CFDP Events](#)

1.9.1 CFS CFDP Telemetry Descriptions

CF Housekeeping Telemetry Packet

The Housekeeping Telemetry packet is sent by CF to the software bus on command. When CF receives the CF_SE \leftarrow ND_HK_MID command, a packet is constructed and sent by CF. CF typically receives this command every four or five seconds.

CF End of Transaction Packet

The End of Transaction packet is sent to the software bus upon completion of a file send or receive transaction. A packet with message ID [CF_EOT_TLM_MID](#) is constructed and sent by CF. The packet contains information about the last completed transaction which includes sequence number, channel, direction, state, status, EID, file size, CRC result, and filenames.

Prev: [CFS CFDP Telemetry](#)

Next: [CFS CFDP Table Definitions](#)

1.10 CFS CFDP Events

[CFS CFDP Event IDs](#)

Prev: [CFS CFDP Telemetry](#)

Next: [CFS CFDP Constraints](#)

1.11 CFS CFDP Constraints

CF is currently limited by the Software Bus performance. Future work may include implementing a point-to-point interface for high speed transfers.

The CFDP standard CCSDS 727.0-B-5 allows a variety of data-type sizes for Source and Destination Entity ID's, Transaction ID and PDU Header length. Currently, this application does not support the following:

1. an Entity ID length other than the size defined in cf_platform_cfg.h
2. a transaction ID length other than the size defined in cf_platform_cfg.h
3. a PDU header size greater than CF_CFDP_MAX_HEADER_SIZE bytes

The stack size for the CF application must be monitored and must be no less than 16384 bytes. Depending on the CF configuration, the stack size may need to be set higher than 16384. The stack size is specified in the cfe_es_startup.scr file that is located in the /mission/build/xxx/exe area.

Poll directories must not have subdirectories, otherwise errors will occur at put request time.

All files in polling directories must be closed.

The same file cannot be sent on two channels at same time.

Spaces are not allowed in filenames.

Segmentation control as described in the blue book is not supported by this version of CFS CF application. All outgoing transactions will have the segmentation control bit in the meta-data PDU set to - 'Recorded Boundaries Not Respected'.

This version of CF does not support keep alive procedures that are detailed in section 4.1.6.5 of CFDP CCSDS 727.0-B-5 Blue Book.

Invalid file structures are not supported by this version of the CF application. Invalid file structures are described in 4.1.6.1.8 of CFDP CCSDS 727.0-B-5 Blue Book and apply only when record boundaries are respected (see segmentation control above).

Only transmission modes unacknowledged (class 1) and acknowledged (class 2) are supported by the CF application.

The CFDP Application will fail on startup if the following conditions are not met:

- Unable to create a Software Bus Pipe

- Unable to subscribe to the CF Command Message
- Unable to subscribe to the CF Housekeeping Request Message
- Unable to register for cFE Event Services
- Unable to register the CF Configuration Table with cFE Table Services
- Unable to load the CF Configuration Table with a default table file
- Unable to acquire a pointer to the CF Configuration Table

Each one of these conditions will generate a unique event message and will cause the CF Application to terminate before processing any CF command pipe messages.

Prev: [CFS CFDP Events](#)

Next: [CFS CFDP Frequently Asked Questions](#)

1.12 CFS CFDP Frequently Asked Questions

No CF specific FAQ's have been identified/documentated.

Prev: [CFS CFDP Constraints](#)

Next: [CFS CFDP Lessons Learned](#)

1.13 CFS CFDP Lessons Learned

These are the lessons learned setting up CF application:

- If you get a "PDU too short":
 - The ground system might not be setting the "length" in the CCSDS header correctly.
- If you get an "EOF PUD too short" or missing 2 bytes from an uploaded file:
 - The current version of CF does not support PDU CRC. The ground system needs to set the "CRC Flag" to false.
- If you get a "CRC mismatch for R trans":
 - This is referencing the "checksum" and not the "PDU CRC".
 - Make sure you are using the modular checksum calculation (Type 0) in your ground system.
 - The PDU can be truncated. Check the "max PDU size" in your CF application and the "max ingest" of the CI application (increase as necessary). Otherwise set the "max file segment" smaller in the ground system.
- If you get an "inactivity timer expired" after the ground system sends an EOF PDU:
 - If you are sending the file in acknowledge mode, it could be the checksum calculation is taking longer than your "inactivity timer". You can increase the "inactivity timer" or increase the "max number of bytes per wakeup to calculate CRC".

2 Core Flight Executive Documentation

- General Information and Concepts
 - Background
 - Applicable Documents
 - Version Numbers
 - Dependencies
 - Acronyms
 - Glossary of Terms
- 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)
 - cFE Time Services Overview
 - cFE Time Services Commands
 - cFE Time Services Telemetry
 - TIME Event Message Reference
 - cFE Time Services Configuration Parameters

- [cFE Event Message Cross Reference](#)
- [cFE Command Mnemonic Cross Reference](#)
- [cFE Telemetry Mnemonic Cross Reference](#)
- [cFE Application Programmer's Interface \(API\) Reference](#)

2.1 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.

2.1.1 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.

2.1.1.1 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.

2.2 Applicable Documents

Document Title	Link
cFE System (L4) Requirements Document	cfe/docs/cfe_requirements.docx'
cFE Functional (L5) Requirements Document	cfe/docs/cFE_FunctionalRequirements.csv
cFE Application Developers Guide	cfe/docs/cFE_Application_Developers_Guide.md'
cFE User's Guide (includes API)	Autogenerated from code, provided with releases in cFE repository
OS Abstraction Layer (OSAL) API	Autogenerated from code, provided with releases in OSAL repository

2.3 Version Numbers

2.3.1 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 Revision number, and the Mission Revision number.

It is important to note that version numbers are only updated upon official releases of tagged versions, **NOT** on development builds. We aim to follow the Semantic Versioning v2.0 specification with our versioning.

The MAJOR number is incremented on release to indicate when there is a change to an API that may 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 is incremented on release to indicate the addition of features to the API which do not break the 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 REVISION number shall be incremented on changes that benefit from unique identification such as bug fixes or major documentation updates. The Revision number 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. WARNING: The revision number is set to the number 99 in development builds. To distinguish between development builds refer to the BUILD_NUMBER and BUILD_BASELINE detailed in the section "Identifying Development Builds".

The Mission Rev Version number is set to zero in all official releases, and is reserved for the mission use.

2.3.2 How and Where Defined

The version numbers 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.

Note the Mission Rev number is provided for missions to be able to identify unique changes they have made to the released software (via clone and own). Specifically, the values 1-254 are reserved for mission use to denote patches/customizations while 0 and 0xFF are reserved for cFS open-source development use (pending resolution of nasa/cFS#440).

2.3.3 Identifying Development Builds

In order to distinguish between development versions, we also provide a BUILD_NUMBER.

The BUILD_NUMBER reflects the number of commits since the BUILD_BASELINE, a baseline git tag, for each particular component. The BUILD_NUMBER integer monotonically increases for a given baseline. The BUILD_BASELINE identifies the current development cycle and is a git tag with format vMAJOR.MINOR.REVISION. The Codename used in the version string also refers to the current development cycle. When a new baseline tag and codename are created, the BUILD_NUMBER resets to zero and begins increasing from a new baseline.

2.3.4 Templates for the short and long version string

See [cfe_version.h](#) for the standard layout and definition of version information. The apps and repositories follow the same pattern by replacing the CFE_ prefix with the appropriate name; for example, osal uses OS_, psp uses CFE_P←SP_IMPL, and so on.

Suggested pattern for development:

- CFSCOMPONENT_SRC_VERSION: REFERENCE_GIT_TAG"+dev"BUILD_NUMBER
 - Example: "v6.8.0-rc1+dev123"
- CFSCOMPONENT_VERSION_STRING: "CFSCOMPONENT DEVELOPMENT BUILD "CFSCOMPONENT_S←RC_VERSION" (Codename: CFSCONSTELLATION), Last Official Release: MAJOR.MINOR.REVISION"
 - Example: "cFE DEVELOPMENT BUILD v6.8.0-rc1+dev123 (Codename: Bootes), Last Official Release: cfe v6.7.0"

Suggested pattern for official releases:

- CFSCOMPONENT_SRC_VERSION: OFFICIAL_GIT_TAG
 - Example: "v7.0.0"
- COMPONENT_VERSION_STRING: "CFSCOMPONENT OFFICIAL RELEASE "CFSCOMPONENT_SRC_VERSION" (Codename: CFSCONSTELLATION)"
 - Example: "cFE OFFICIAL RELEASE v7.0.0 (Codename: Caelum)"

2.4 Dependencies

The Core Flight Executive (cFE) is required to be built with the Operating System Abstraction Layer (OSAL) and Platform Support Package (PSP) components of the Core Flight System (cFS). It is always recommended to build with the latest versions of each of the components as backward compatibility 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**.

2.5 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

Acronym	Description
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

2.6 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 (OSAL) and platform through the Platform Support Package (PSP).

The functionality provided by the ES task include Software Reset, Application and Child Task Management, Basic File System, Performance Data Collection, Critical Data Store, Memory Pool, System Log, Shell Command.

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

- [Terminology](#)
- [Software Reset](#)
 - [Reset Types and Subtypes](#)
 - [Exception and Reset \(ER\) Log](#)

- Application and Child Task Management
 - Starting an Application
 - Stopping an Application
 - Restarting an Application
 - Reloading an Application
 - Listing Current Applications
 - Listing Current Tasks
 - Loading Common Libraries
- Basic File System
- Performance Data Collection
- Critical Data Store
- Memory Pool
- System Log
- Version Identification
- Frequently Asked Questions about Executive Services

2.6.1 Terminology

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

- "Application" and "cFE Application"
- "Task"
- "Startup Script"

2.6.1.1 "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.

cFE 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 'Service' or 'Core Application' is typically 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.

2.6.1.2 "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.

2.6.1.3 "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. For a processor reset, ES checks for the CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE first, and if it doesn't exist or for a power on reset ES uses the file passed in to [CFE_ES_Main](#) (typically CFE_PLATFORM_ES_NONVOL_STARTUP_FILE but dependent on the PSP).

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	<p>This is the Action the cFE should take if the Application has an exception.</p> <ul style="list-style-type: none"> • 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_PLATFORM_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_PLATFORM_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](#). The [System Log](#) may also contain positive acknowledge messages regarding the startup script processing.

The startup script delivered with the cFE (`cfe_es_startup.scr`) also has some detailed information about the fields and the settings.

2.6.2 Software Reset

The ES Software Reset 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 include 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 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](#).

2.6.3 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_PSP_RST_TYPE_POWERON](#) and [CFE_PSP_RST_TYPE_PROCESSOR](#). 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.

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_PSP_RST_SUBTYPE_POWER_CYCLE](#), [CFE_PSP_RST_SUBTYPE_PUSH_BUTTON](#), [CFE_PSP_RST_SUBTYPE_HW_SPECIA](#), [CFE_PSP_RST_SUBTYPE_HW_WATCHDOG](#), [CFE_PSP_RST_SUBTYPE_RESET_COMMAND](#), [CFE_PSP_RST_SUBTYPE_EXCEP](#), [CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET](#), [CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET](#), [CFE_PSP_RST_SUBTYPE_BAN](#)

2.6.4 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_PLATFORM_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_PLATFORM_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_PLATFORM_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](#).

2.6.5 Application and Child Task Management

The ES Application and Child Task Management provides 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 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.

2.6.6 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.

The format of the Start Application command, is defined in the structure [CFE_ES_StartAppCmd_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, file-name 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.

2.6.7 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_StopAppCmd_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.

2.6.8 Restarting an Application

The [CFE_ES_RESTART_APP_CC](#) command is used to restart an application using the same file name as the last start.

This command checks for file existence, the application is running, and the application is not a core app. If valid, the application restart is requested.

When requested, ES stops the application, unloads the object file, loads the object file using the previous file name, and restarts an application using the parameters defined when the application was previously started, either through the startup script or by way of the [CFE_ES_START_APP_CC](#) command.

2.6.9 Reloading an Application

The [CFE_ES_RELOAD_APP_CC](#) command is used to reload an application using a new file name.

This command performs the same actions as [CFE_ES_RESTART_APP_CC](#) only using the new file.

2.6.10 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 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](#).

2.6.11 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

2.6.12 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.

2.6.13 Basic File System

ES provides minimal functionality to initialize, read, and write cfe File headers.

2.6.14 Performance Data Collection

The Performance Data Collection provides precise timing information for each software application similar to how a logic analyzer can trigger and filter data.

API calls 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 analysis tools for viewing. The size of the buffer is configurable through the [CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE](#) platform configuration parameter.

Additional information follows:

- [Performance Data Collection Trigger Masks](#)
- [Starting to Collect Performance Data](#)
- [Stopping the Collection of Performance Data](#)
- [Viewing the Collection of Performance Data](#)

2.6.14.1 Performance Data Collection Trigger Masks The trigger mask is used to control 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 to begin 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.

2.6.14.2 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.

2.6.14.3 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_PLATFORM_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.

2.6.14.4 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 a viewing tool. See <https://github.com/nasa/perfutils-java> as an example.

2.6.15 Critical Data Store

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 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](#)).

2.6.16 Memory Pool

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_GetPoolBuf](#) 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 ensures 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_MISSION_ES_POOL_MAX_BUCKETS](#)), 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

2.6.17 System Log

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_SYS_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_PLATFORM_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_SYS_LOG_CC](#) is used to clear the System log.

The size of the System log is defined by the platform configuration parameter [CFE_PLATFORM_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.

2.6.18 Version Identification

Version information is reported at startup, and upon receipt of a No-op command

2.6.19 Frequently Asked Questions about Executive Services

None submitted

2.7 cFE Executive Services Commands

Upon receipt of any command, the Executive Services application will confirm that the message length embedded within the header (from `CFE_MSG_GetSize()`) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, ES will generate the `CFE_ES_LEN_ERR_EID` event, increment the command error counter (`$sc_$cpu_ES_CMDEC`), and the command will *not* be accepted for processing.

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 Exception and Reset Log

Global `CFE_ES_CLEAR_SYS_LOG_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_SYS_LOG_CC`

Set Executive Services System Log Mode to Discard/Overwrite

Global `CFE_ES_QUERY_ALL_CC`

Writes all Executive Services Information on all loaded modules 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 module

Global `CFE_ES_RELOAD_APP_CC`

Stops, Unloads, Loads from the command specified File and Restarts an Application

Global `CFE_ES_RESET_COUNTERS_CC`

Executive Services Reset Counters

Global `CFE_ES_RESET_PR_COUNT_CC`

Resets the Processor Reset Counter to Zero

Global `CFE_ES_RESTART_APP_CC`

Stops, Unloads, Loads using the previous File name, and Restarts an Application

Global `CFE_ES_RESTART_CC`

Executive Services Processor / Power-On Reset

Global `CFE_ES_SEND_MEM_POOL_STATS_CC`

Telemeter Memory Pool Statistics

Global `CFE_ES_SET_MAX_PR_COUNT_CC`

Configure the Maximum Number of Processor Resets before a Power-On Reset

Global CFE_ES_SET_PERF_FILTER_MASK_CC

Set Performance Analyzer's Filter Masks

Global CFE_ES_SET_PERF_TRIGGER_MASK_CC

Set Performance Analyzer's Trigger Masks

Global CFE_ES_START_APP_CC

Load and Start an Application

Global CFE_ES_START_PERF_DATA_CC

Start Performance Analyzer

Global CFE_ES_STOP_APP_CC

Stop and Unload Application

Global CFE_ES_STOP_PERF_DATA_CC

Stop Performance Analyzer and write data file

Global CFE_ES_WRITE_ER_LOG_CC

Writes Exception and Reset Log to a File

Global CFE_ES_WRITE_SYS_LOG_CC

Writes contents of Executive Services System Log to a File

2.8 cFE Executive Services Telemetry

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

Global CFE_ES_HousekeepingTlm_Payload_t

Executive Services Housekeeping Packet

Global CFE_ES_HousekeepingTlm_Payload_t

Executive Services Housekeeping Packet

Global CFE_ES_HousekeepingTlm_t

Executive Services Housekeeping Packet

Global CFE_ES_HousekeepingTlm_t

Executive Services Housekeeping Packet

Global CFE_ES_MemStatsTlm_t

Memory Pool Statistics Packet

Global CFE_ES_MemStatsTlm_t

Memory Pool Statistics Packet

Global CFE_ES_OneAppTlm_Payload_t

Single Application Information Packet

Global CFE_ES_OneAppTlm_Payload_t

Single Application Information Packet

Global CFE_ES_OneAppTlm_t

Single Application Information Packet

Global CFE_ES_OneAppTlm_t

Single Application Information Packet

Global CFE_ES_PoolStatsTlm_Payload_t

Memory Pool Statistics Packet

Global CFE_ES_PoolStatsTlm_Payload_t

Memory Pool Statistics Packet

2.9 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_FULL_NAME_LEN

Maximum Length of Full CDS Name in messages

Maximum Length of Full CDS Name in messages

Global CFE_MISSION_ES_CDS_MAX_NAME_LENGTH

Maximum Length of CDS Name

Maximum Length of CDS Name

Global CFE_MISSION_ES_DEFAULT_CRC

Mission Default CRC algorithm

Mission Default CRC algorithm

Global CFE_MISSION_ES_MAX_APPLICATIONS

Mission Max Apps in a message

Mission Max Apps in a message

Global CFE_MISSION_ES_PERF_MAX_IDS

Define Max Number of Performance IDs for messages

Define Max Number of Performance IDs for messages

Global CFE_MISSION_ES_POOL_MAX_BUCKETS

Maximum number of block sizes in pool structures

Maximum number of block sizes in pool structures

Global CFE_PLATFORM_CORE_MAX_STARTUP_MSEC

CFE core application startup timeout

Global CFE_PLATFORM_ES_APP_KILL_TIMEOUT

Define ES Application Kill Timeout

Define ES Application Kill Timeout

Global CFE_PLATFORM_ES_APP_SCAN_RATE

Define ES Application Control Scan Rate

Define ES Application Control Scan Rate

Global CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES

Define Maximum Number of Registered CDS Blocks

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

Define ES Critical Data Store Memory Pool Block Sizes

Global CFE_PLATFORM_ES_CDS_SIZE

Define Critical Data Store Size

Define Critical Data Store Size

Global CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE

Default Application Information Filename

Default Application Information Filename

Global CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE

Default Critical Data Store Registry Filename

Default Critical Data Store Registry Filename

Global CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE

Default Exception and Reset (ER) Log Filename

Default Exception and Reset (ER) Log Filename

Global CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME

Default Performance Data Filename

Default Performance Data Filename

Global CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE

Define Default System Log Mode following Power On Reset

Define Default System Log Mode following Power On Reset

Global CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE

Define Default System Log Mode following Processor Reset

Define Default System Log Mode following Processor Reset

Global CFE_PLATFORM_ES_DEFAULT_STACK_SIZE

Define Default Stack Size for an Application

Define Default Stack Size for an Application

Global CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE

Default System Log Filename

Default System Log Filename

Global CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE

Default Application Information Filename

Default Application Information Filename

Global CFE_PLATFORM_ES_ER_LOG_ENTRIES

Define Max Number of ER (Exception and Reset) log entries

Define Max Number of ER (Exception and Reset) log entries

Global CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE

Maximum size of CPU Context in ES Error Log

Maximum size of CPU Context in ES Error Log

Global CFE_PLATFORM_ES_MAX_APPLICATIONS

Define Max Number of Applications

Define Max Number of Applications

Global CFE_PLATFORM_ES_MAX_GEN_COUNTERS

Define Max Number of Generic Counters

Define Max Number of Generic Counters

Global CFE_PLATFORM_ES_MAX_LIBRARIES

Define Max Number of Shared libraries

Define Max Number of Shared libraries

Global CFE_PLATFORM_ES_MAX_MEMORY_POOLS

Maximum number of memory pools

Maximum number of memory pools

Global CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS

Define Number of Processor Resets Before a Power On Reset

Define Number of Processor Resets Before a Power On Reset

Global CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01

Define Default ES Memory Pool Block Sizes

Define Default ES Memory Pool Block Sizes

Global CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN

Define Memory Pool Alignment Size

Define Memory Pool Alignment Size

Global CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING

Default virtual path for persistent storage

Default virtual path for persistent storage

Global CFE_PLATFORM_ES_NONVOL_STARTUP_FILE

ES Nonvolatile Startup Filename

ES Nonvolatile Startup Filename

Global CFE_PLATFORM_ES_OBJECT_TABLE_SIZE

Define Number of entries in the ES Object table

Define Number of entries in the ES Object table

Global CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY

Define Performance Analyzer Child Task Delay

Define Performance Analyzer Child Task Delay

Global CFE_PLATFORM_ES_PERF_CHILD_PRIORITY

Define Performance Analyzer Child Task Priority

Define Performance Analyzer Child Task Priority

Global CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE

Define Performance Analyzer Child Task Stack Size

Define Performance Analyzer Child Task Stack Size

Global CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE

Define Max Size of Performance Data Buffer

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

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

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

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
Define Filter Mask Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_TRIGMASK_ALL

Define Filter Trigger Setting for Enabling All Performance Entries
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
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
Define Default Filter Trigger Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_POOL_MAX_BUCKETS

Maximum number of block sizes in pool structures
Maximum number of block sizes in pool structures

Global CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING

Default virtual path for volatile storage
Default virtual path for volatile storage

Global CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS

ES Ram Disk Number of Sectors
ES Ram Disk Number of Sectors

Global CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED

Percentage of Ram Disk Reserved for Decompressing Apps
Percentage of Ram Disk Reserved for Decompressing Apps

Global CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE

ES Ram Disk Sector Size
ES Ram Disk Sector Size

Global CFE_PLATFORM_ES_START_TASK_PRIORITY

Define ES Task Priority
Define ES Task Priority

Global CFE_PLATFORM_ES_START_TASK_STACK_SIZE

Define ES Task Stack Size
Define ES Task Stack Size

Global CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC

Startup script timeout
Startup script timeout

Global CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC

Poll timer for startup sync delay
Poll timer for startup sync delay

Global CFE_PLATFORM_ES_SYSTEM_LOG_SIZE

Define Size of the cFE System Log.
Define Size of the cFE System Log.

Global CFE_PLATFORM_ES_USER_RESERVED_SIZE

Define User Reserved Memory Size

Define User Reserved Memory Size

Global CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE

ES Volatile Startup Filename

ES Volatile Startup Filename

Global CFE_PLATFORM_EVS_START_TASK_PRIORITY

Define EVS Task Priority

Define EVS Task Priority

Global CFE_PLATFORM_EVS_START_TASK_STACK_SIZE

Define EVS Task Stack Size

Define EVS Task Stack Size

Global CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01

Define SB Memory Pool Block Sizes

Define SB Memory Pool Block Sizes

Global CFE_PLATFORM_SB_START_TASK_PRIORITY

Define SB Task Priority

Define SB Task Priority

Global CFE_PLATFORM_SB_START_TASK_STACK_SIZE

Define SB Task Stack Size

Define SB Task Stack Size

Global CFE_PLATFORM_TBL_START_TASK_PRIORITY

Define TBL Task Priority

Define TBL Task Priority

Global CFE_PLATFORM_TBL_START_TASK_STACK_SIZE

Define TBL Task Stack Size

Define TBL Task Stack Size

2.10 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 from within the context of a registered application or core service. EVS provides various ways to filter event messages in order to manage event message generation.

Note for messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE_ES_WriteToSysLog](#) can be used for reporting.

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
- EVS squelching of misbehaving apps
- Frequently Asked Questions about Event Services

2.10.1 Event Message Format

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

- Timestamp
- Event Type
- Spacecraft ID
- Processor ID
- Application Name
- Event ID
- Message

The *Timestamp* corresponds to when the event was generated, in spacecraft time. The *Event Type* is one of the following: 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 *Application Name* refers to the Application that issued the event message as specified on application startup (either startup script or app start command). The *Event ID* is an Application unique number that identifies the event. 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.

2.10.2 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](#).

2.10.2.1 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](#).

2.10.3 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.

2.10.3.1 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 messages 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_PLATFORM_EVS_DEFAULT_TYPE_FLAG](#) in the cfe_platform_cfg.h file specifies which event message types are enabled/disabled by default.

2.10.3.2 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).

2.10.3.3 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.

2.10.3.4 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.

2.10.3.4.1 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_PLATFORM_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.

2.10.3.4.2 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 an 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.

2.10.4 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.

2.10.5 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_PLATFORM_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

2.10.6 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.

2.10.7 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.

2.10.8 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.

2.10.9 EVS squelching of misbehaving apps

Event squelching is an optional feature for suppressing excessive events from misbehaving apps. It is enabled by setting `CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST` to a nonzero positive value, and `CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC` equal to or less than that value.

`CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST` controls the maximum events that can be sent at a given moment, and `CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC` is the sustained event throughput per second.

The suppression mechanism initializes with `CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST * 1000` credits. Each event costs 1000 credits. Credits are restored at a rate of `CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC * 1000` up to a maximum balance of `CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST * 1000`, and the maximum "debt" is `-CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST * 1000`. When the credit count crosses from positive to negative, a squelched event message is emitted and events are suppressed, until the credit count becomes positive again.

Figure EVS-1 is a notional state diagram of the event squelching mechanism.

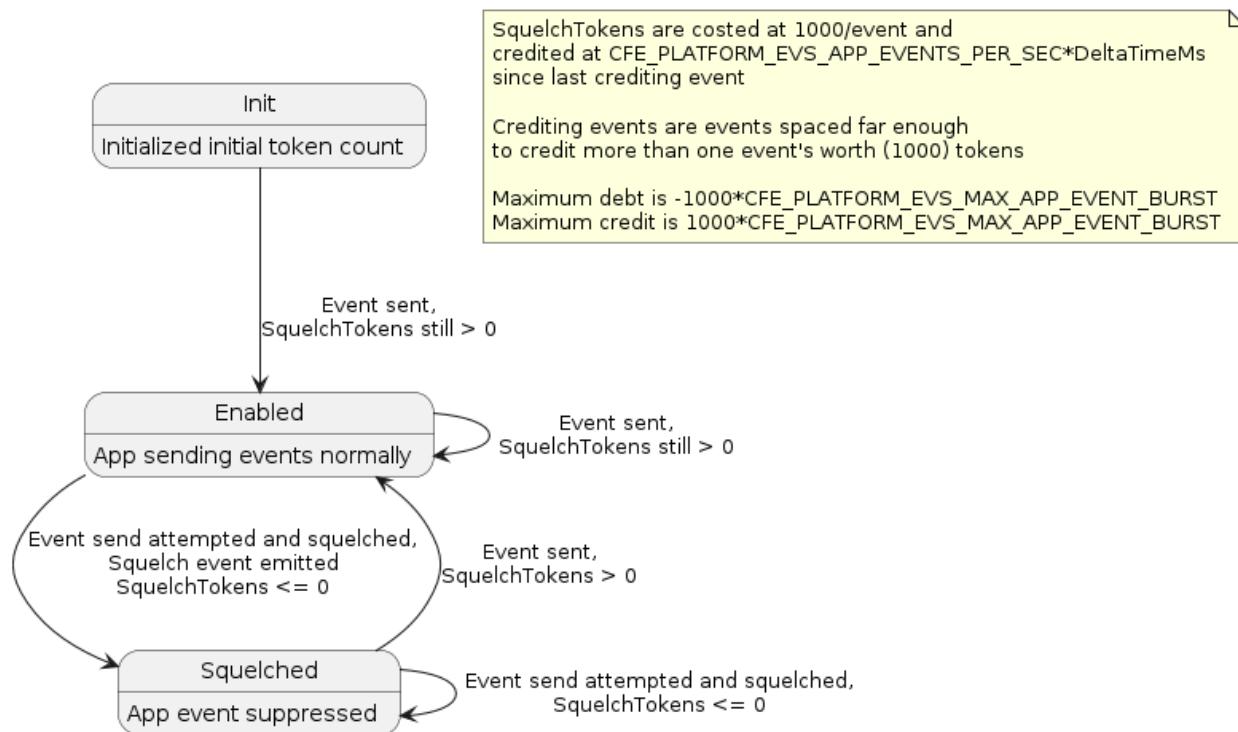


Figure 1 Figure EVS-1: EVS Squelching State Diagram

2.10.10 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*
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.

(Q) Why do I see event messages in my console window?

By default, the events are configured to transmit out a "port" that shows event messages in the console

(Q) What is the difference between event services and the ES System Log

Events are within the context of an App or cFE Service (requires registration with ES). The system log can be written to outside of the Application or cFE Service context, for example during application startup to report errors before registration.

2.11 cFE Event Services Commands

Upon receipt of any command, the Event Services application will confirm that the message length embedded within the header (from [CFE_MSG_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, EVS will generate the [CFE_EVS_LEN_ERR_EID](#) event, increment the command error counter ([\\$sc_\\$cpu_EVS_CMDEC](#)), and the command will *not* be accepted for processing.

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

Global CFE_EVS_DISABLE_APP_EVENTS_CC

Disable Event Services for an Application

Global CFE_EVS_DISABLE_EVENT_TYPE_CC

Disable Event Type

Global CFE_EVS_DISABLE_PORTS_CC

Disable Event Services Output Ports

Global CFE_EVS_ENABLE_APP_EVENT_TYPE_CC

Enable Application Event Type

Global CFE_EVS_ENABLE_APP_EVENTS_CC

Enable Event Services for an Application

Global CFE_EVS_ENABLE_EVENT_TYPE_CC

Enable Event Type

Global CFE_EVS_ENABLE_PORTS_CC

Enable Event Services Output Ports

Global CFE_EVS_NOOP_CC

Event Services No-Op

Global CFE_EVS_RESET_ALL_FILTERS_CC

Reset All Event Filters for an Application

Global CFE_EVS_RESET_APP_COUNTER_CC

Reset Application Event Counters

Global CFE_EVS_RESET_COUNTERS_CC

Event Services Reset Counters

Global CFE_EVS_RESET_FILTER_CC

Reset an Event Filter for an Application

Global CFE_EVS_SET_EVENT_FORMAT_MODE_CC

Set Event Format Mode

Global CFE_EVS_SET_FILTER_CC

Set Application Event Filter

Global CFE_EVS_SET_LOG_MODE_CC

Set Logging Mode

Global CFE_EVS_WRITE_APP_DATA_FILE_CC

Write Event Services Application Information to File

Global CFE_EVS_WRITE_LOG_DATA_FILE_CC

Write Event Log to File

2.12 cFE Event Services Telemetry

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

Global `CFE_EVS_HousekeepingTlm_Payload_t`

Event Services Housekeeping Telemetry Packet

Global `CFE_EVS_HousekeepingTlm_Payload_t`

Event Services Housekeeping Telemetry Packet

Global `CFE_EVS_LongEventTlm_Payload_t`

Event Message Telemetry Packet (Long format)

Global `CFE_EVS_LongEventTlm_Payload_t`

Event Message Telemetry Packet (Long format)

Global `CFE_EVS_ShortEventTlm_Payload_t`

Event Message Telemetry Packet (Short format)

Global `CFE_EVS_ShortEventTlm_Payload_t`

Event Message Telemetry Packet (Short format)

2.13 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

Maximum Event Message Length

Global `CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC`

Sustained number of event messages per second per app before squelching

Sustained number of event messages per second per app before squelching

Global `CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE`

Default EVS Application Data Filename

Default EVS Application Data Filename

Global `CFE_PLATFORM_EVS_DEFAULT_LOG_FILE`

Default Event Log Filename

Default Event Log Filename

Global `CFE_PLATFORM_EVS_DEFAULT_LOG_MODE`

Default EVS Local Event Log Mode

Default EVS Local Event Log Mode

Global `CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE`

Default EVS Message Format Mode

Default EVS Message Format Mode

Global CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG

Default EVS Event Type Filter Mask

Default EVS Event Type Filter Mask

Global CFE_PLATFORM_EVS_LOG_MAX

Maximum Number of Events in EVS Local Event Log

Maximum Number of Events in EVS Local Event Log

Global CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST

Maximum number of event before squelching

Maximum number of event before squelching

Global CFE_PLATFORM_EVS_MAX_EVENT_FILTERS

Define Maximum Number of Event Filters per Application

Define Maximum Number of Event Filters per Application

Global CFE_PLATFORM_EVS_PORT_DEFAULT

Default EVS Output Port State

Default EVS Output Port State

2.14 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 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.

- [Software Bus Terminology](#)
- [Autonomous Actions](#)
- [Operation of the SB Software](#)
- [Frequently Asked Questions about Software Bus](#)

2.14.1 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](#)

2.14.1.1 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) in [CCSDS Space Packet Protocol](#), but can be customized by replacing the message module.

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).

The concept of a message identifier is utilized to provide abstraction from header implementation, often abbreviated as message ID, MsgId, or MID. Header and message identifier values should not be accessed directly to avoid implementation specific dependencies.

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 message module provides APIs for 'setting' and 'getting' the fields in the header of the message. The message module was separated from software bus to enable users to customize message headers without requiring clone and own of the entire cfe repository. To customize, remove the built in msg module from the build and replace with custom implementation. See sample target definitions folder for examples.

Following the header is the user defined message data.

2.14.1.2 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 ['Write 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](#).

2.14.1.3 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_PLATFORM_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.

2.14.1.4 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_PLATFORM_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_PLATFORM_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_ReceiveBuffer for the pipe that received the buffer.

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_PLATFORM_SB_BUF_MEMORY_BYTES](#) minus 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.

2.14.2 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.

2.14.3 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

2.14.3.1 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.

2.14.3.2 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.

2.14.3.3 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.

2.14.3.4 Packet Sequence Values The sequence count behavior depends on if the message is a command type or telemetry type.

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

For a telemetry message, the behavior is controlled via API input parameters when sending. When enabled, the software bus will populate the packet sequence counter using an internal counter that gets initialized upon the first subscription to the message (first message will have a packet sequence counter value of 1). From that point on each send request will increment the counter by one, regardless of the number of destinations or if there is an active subscription.

After a rollover condition the sequence counter will be a value of zero for one instance. The sequence counter is incremented after all the checks have passed prior to the actual sending of the message. This includes the parameter checks and the memory allocation check.

When disabled, the original message will not be altered. 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.

2.14.3.5 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_PLATFORM_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.

2.14.3.6 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.

2.14.3.7 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 error cases encountered when sending a message generate an event, and events cause a message to be sent 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 software bus needs to send an event that may cause recursion, the flag is set and the event is sent. If sending the event would cause the same event again, the event call will be 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.

2.14.3.8 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.

2.14.3.9 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.

2.14.3.10 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 implement quality of service.

A default quality of services is provided via the [CFE_SB_DEFAULT_QOS](#) macro.

2.14.3.11 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_PLATFORM_SB_BUF_MEMORY_BYTES](#) which indicates the amount allocated.

2.14.4 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](#)

(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 when sending a message (from the callers memory space) using [CFE_SB_TransmitMsg](#). When transmitting the message, the software bus copies the message from the callers memory space into a buffer in the software bus memory space. There is also the option to request a buffer from SB, write directly to the buffer and send via [CFE_SB_TransmitBuffer](#). This is equivalent to the previous zero copy implementation. The [CFE_SB_ReceiveBuffer](#) API gives the user back a pointer to the buffer. When working with the buffers, the additional complexity to be aware of is the buffer is only available to the app from the request to send (on the sending side), or from the receive until the next receive on the same pipe on the receiving side. If the data is required outside that scope, the app needs a local copy.

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

After receiving a buffer by calling [CFE_SB_ReceiveBuffer](#), the buffer received is valid until the next call to [CFE_SB_ReceiveBuffer](#) with the same Pipe Id. If the caller needs the message longer than the next call to [CFE_SB_ReceiveBuffer](#), the caller must copy the message to its memory space.

(Q) The first parameter in the [CFE_SB_ReceiveBuffer](#) 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_Buffer_t` (i.e. `CFE_SB_Buffer_t *Ptr`) then gives the address of that pointer (`&Ptr`) as this parameter. After a successful call to [CFE_SB_ReceiveBuffer](#), `Ptr` will point to the first byte of the software bus buffer. 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_eventids.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 as [CFE_SB_DEFAULT_QOS](#) will ensure seamless integration when the software bus is expanded to support inter-processor communication.

(Q) Can I confirm my software bus buffer was delivered?

There is no built in mechanism for confirming delivery (it could span systems). This could be accomplished by generating a response message from the receiver.

2.15 cFE Software Bus Commands

Upon receipt of any command, the Software Bus application will confirm that the message length embedded within the header (from [CFE_MSG_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, SB will generate the [CFE_SB_LEN_ERR_EID](#) event, increment the command error counter (\$sc_\$cpu_SB_CMDEC), and the command will *not* be accepted for processing.

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_PREV_SUBS_CC

Send Previous Subscriptions Command

Global CFE_SB_SEND_SB_STATS_CC

Send Software Bus Statistics

Global CFE_SB_WRITE_MAP_INFO_CC

Write Map Info to a File

Global CFE_SB_WRITE_PIPE_INFO_CC

Write Pipe Info to a File

Global CFE_SB_WRITE_ROUTING_INFO_CC

Write Software Bus Routing Info to a File

2.16 cFE Software Bus Telemetry

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

Global CFE_SB_AllSubscriptionsTlm_Payload_t

SB Previous Subscriptions Packet

Global CFE_SB_AllSubscriptionsTlm_Payload_t

SB Previous Subscriptions Packet

Global CFE_SB_HousekeepingTlm_Payload_t

Software Bus task housekeeping Packet

Global CFE_SB_HousekeepingTlm_Payload_t

Software Bus task housekeeping Packet

Global CFE_SB_SingleSubscriptionTlm_Payload_t

SB Subscription Report Packet

Global CFE_SB_SingleSubscriptionTlm_Payload_t

SB Subscription Report Packet

Global CFE_SB_StatsTlm_Payload_t

SB Statistics Telemetry Packet

Global CFE_SB_StatsTlm_Payload_t

SB Statistics Telemetry Packet

2.17 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

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

Global CFE_MISSION_SB_MAX_SB_MSG_SIZE

Maximum SB Message 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

Size of the SB buffer memory pool

Global CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME

Default Message Map Filename

Default Message Map Filename

Global CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT

Default Subscription Message Limit

Default Subscription Message Limit

Global CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME

Default Pipe Information Filename

Default Pipe Information Filename

Global CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME

Default Routing Information Filename

Default Routing Information Filename

Global CFE_PLATFORM_SB_FILTERED_EVENT1

SB Event Filtering

SB Event Filtering

Global CFE_PLATFORM_SB_HIGHEST_VALID_MSGID

Highest Valid Message Id

Highest Valid Message Id

Global CFE_PLATFORM_SB_MAX_DEST_PER_PKT

Maximum Number of unique local destinations a single MsgId can have

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

Maximum Number of Unique Message IDs SB Routing Table can hold

Global CFE_PLATFORM_SB_MAX_PIPES

Maximum Number of Unique Pipes SB Routing Table can hold

Maximum Number of Unique Pipes SB Routing Table can hold

2.18 cFE Table Services Overview

Applications often 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.

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.

None of the cFE core applications (EVS, SB, ES, TIME, or TBL) use tables, and it is possible to build cFE without Table Services if not needed or an alternative parameter management mechanism is to be utilized.

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

- [Managing Tables](#)
- [cFE Table Types and Table Options](#)
- [Table Registry](#)
- [Table Services Telemetry](#)
- [Effects of Processor Reset on Tables](#)
- [Frequently Asked Questions about Table Services](#)

2.18.1 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.

2.18.2 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](#)

2.18.2.1 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_PLATFORM_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](#)).

2.18.2.2 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_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS](#) 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.

2.18.2.3 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.

2.18.2.4 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.

2.18.2.5 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.

2.18.2.6 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 telemetry 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.

2.18.3 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.

2.18.4 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_HousekeepingTlm_t](#).

2.18.5 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.

2.18.6 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.

2.19 cFE Table Services Commands

Upon receipt of any command, the Table Services application will confirm that the message length embedded within the header (from `CFE_MSG_GetSize()`) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, TBL will generate the `CFE_TBL_LEN_ERR_EID` event, increment the command error counter (`$sc_$cpu_TBL_CMDEC`), and the command will *not* be accepted for processing.

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

2.20 cFE Table Services Telemetry

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

Global `CFE_TBL_HousekeepingTlm_Payload_t`

Table Services Housekeeping Packet

Global `CFE_TBL_HousekeepingTlm_Payload_t`

Table Services Housekeeping Packet

Global `CFE_TBL_TblRegPacket_Payload_t`

Table Registry Info Packet

Global `CFE_TBL_TblRegPacket_Payload_t`

Table Registry Info Packet

2.21 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
Maximum Length of Full Table Name in messages

Global CFE_MISSION_TBL_MAX_NAME_LENGTH

Maximum Table Name Length
Maximum Table Name Length

Global CFE_PLATFORM_TBL_BUF_MEMORY_BYTES

Size of Table Services Table Memory Pool
Size of Table Services Table Memory Pool

Global CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE

Default Filename for a Table Registry Dump
Default Filename for a Table Registry Dump

Global CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES

Maximum Number of Critical Tables that can be Registered
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
Maximum Size Allowed for a Double Buffered Table

Global CFE_PLATFORM_TBL_MAX_NUM_HANDLES

Maximum Number of Table Handles
Maximum Number of Table Handles

Global CFE_PLATFORM_TBL_MAX_NUM_TABLES

Maximum Number of Tables Allowed to be Registered
Maximum Number of Tables Allowed to be Registered

Global CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS

Maximum Number of Simultaneous Table Validations
Maximum Number of Simultaneous Table Validations

Global CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS

Maximum Number of Simultaneous Loads to Support
Maximum Number of Simultaneous Loads to Support

Global CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE

Maximum Size Allowed for a Single Buffered Table
Maximum Size Allowed for a Single Buffered Table

Global CFE_PLATFORM_TBL_VALID_PRID_1

Processor ID values used for table load validation
Processor ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_PRID_COUNT

Number of Processor ID's specified for validation

Number of Processor ID's specified for validation

Global CFE_PLATFORM_TBL_VALID_SCID_1

Spacecraft ID values used for table load validation

Spacecraft ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_SCID_COUNT

Number of Spacecraft ID's specified for validation

Number of Spacecraft ID's specified for validation

2.22 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
- Time State
- Initialization
 - Power-On Reset
 - Processor Reset
- Initialization
 - Power-On Reset
 - Processor Reset
- Normal Operation
 - Client
 - Server
 - * Setting Time
 - * Adjusting Time
 - * Setting MET
- Frequently Asked Questions about Time Services

2.22.1 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 (G↔PS 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.

2.22.2 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 typically the 32 bits of seconds and the upper 16 bits of subseconds are used for time stamping Software bus messages, but this is dependent on the underlying definition.

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;
```

2.22.3 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:

$$\text{TAI} = \text{MET} + \text{STCF}$$

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:

$$\text{UTC} = \text{TAI} - \text{Leap Seconds}.$$

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:

$$\text{UTC} = \text{MET} + \text{STCF} - \text{Leap Seconds}$$

2.22.4 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](#)

2.22.4.1 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_MISSION_TIME_CFG_DEFAULT_TAI  TRUE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC  FALSE
```

or

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI  FALSE  
#define CFE_MISSION_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_MISSION_TIME_CFG_DEFAULT_TAI](#), [CFE_MISSION_TIME_CFG_DEFAULT_UTC](#)

2.22.4.2 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_MISSION_TIME_CFG_FAKE_TONE    TRUE
```

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

See also

[CFE_MISSION_TIME_CFG_FAKE_TONE](#)

2.22.4.3 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_MISSION_TIME_AT_TONE_WAS
#define CFE_MISSION_TIME_AT_TONE_WILL_BE
```

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

See also

[CFE_MISSION_TIME_AT_TONE_WAS](#), [CFE_MISSION_TIME_AT_TONE_WILL_BE](#)

2.22.4.4 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_MISSION_TIME_MIN_ELAPSED  0
#define CFE_MISSION_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_MISSION_TIME_MIN_ELAPSED](#), [CFE_MISSION_TIME_MAX_ELAPSED](#)

2.22.4.5 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_PLATFORM_TIME_CFG_SERVER    TRUE  
#define CFE_PLATFORM_TIME_CFG_CLIENT   FALSE
```

or

```
#define CFE_PLATFORM_TIME_CFG_SERVER    FALSE  
#define CFE_PLATFORM_TIME_CFG_CLIENT   TRUE
```

See also

[CFE_PLATFORM_TIME_CFG_SERVER](#), [CFE_PLATFORM_TIME_CFG_CLIENT](#)

2.22.4.6 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
```

2.22.4.7 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

2.22.4.8 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_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET    TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS    FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME   FALSE
```

configuration definitions for the particular source.

If the cfe_platform_cfg.h file contains "#define CFE_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET    TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS    FALSE
#define CFE_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET](#), [CFE_PLATFORM_TIME_CFG_SRC_GPS](#), [CFE_PLATFORM_TIME_CFG_SRC_TIME](#)

2.22.4.9 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_PLATFORM_TIME_CFG_SIGNAL  TRUE
```

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

See also

[CFE_PLATFORM_TIME_CFG_SIGNAL](#)

2.22.5 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_MISSION_TIME_CFG_DEFAULT_TAI TRUE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC FALSE
```

or

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI FALSE  
#define CFE_MISSION_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_MISSION_TIME_CFG_DEFAULT_TAI](#), [CFE_MISSION_TIME_CFG_DEFAULT_UTC](#)

2.22.6 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_MISSION_TIME_CFG_FAKE_TONE TRUE
```

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

See also

[CFE_MISSION_TIME_CFG_FAKE_TONE](#)

2.22.7 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_MISSION_TIME_AT_TONE_WAS  
#define CFE_MISSION_TIME_AT_TONE_WILL_BE
```

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

See also

[CFE_MISSION_TIME_AT_TONE_WAS](#), [CFE_MISSION_TIME_AT_TONE_WILL_BE](#)

2.22.8 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_MISSION_TIME_MIN_ELAPSED 0  
#define CFE_MISSION_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_MISSION_TIME_MIN_ELAPSED](#), [CFE_MISSION_TIME_MAX_ELAPSED](#)

2.22.9 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_PLATFORM_TIME_CFG_SERVER TRUE  
#define CFE_PLATFORM_TIME_CFG_CLIENT FALSE
```

or

```
#define CFE_PLATFORM_TIME_CFG_SERVER FALSE  
#define CFE_PLATFORM_TIME_CFG_CLIENT TRUE
```

See also

[CFE_PLATFORM_TIME_CFG_SERVER](#), [CFE_PLATFORM_TIME_CFG_CLIENT](#)

2.22.10 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
```

2.22.11 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

2.22.12 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_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME FALSE
```

configuration definitions for the particular source.

If the cfe_platform_cfg.h file contains "#define CFE_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS FALSE
#define CFE_PLATFORM_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_PLATFORM_TIME_CFG_SRC_MET](#), [CFE_PLATFORM_TIME_CFG_SRC_GPS](#), [CFE_PLATFORM_TIME_CFG_SRC_TIME](#)

2.22.13 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_PLATFORM_TIME_CFG_SIGNAL TRUE
```

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

See also

[CFE_PLATFORM_TIME_CFG_SIGNAL](#)

2.22.14 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_MISSION_TIME_CFG_DEFAULT_TAI](#) and [CFE_MISSION_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.

2.22.15 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.

2.22.16 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.

2.22.17 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](#)

2.22.17.1 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.

2.22.17.2 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.

2.22.18 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.

2.22.19 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.

2.22.20 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](#)

2.22.20.1 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.

2.22.20.2 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.

2.22.21 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.

2.22.22 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.

2.22.23 Normal Operation

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

- [Client](#)
- [Server](#)

2.22.23.1 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.

2.22.23.2 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](#)

2.22.23.2.1 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](#)

2.22.23.2.2 Adjusting Time The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE_TIME_SET_TIME_CC](#) or explicitly using [CFE_TIME_SET_STCF_CC](#). TIME provides the ability to command a one time adjustment ([CFE_TIME_ADD_ADJUST_CC](#) and [CFE_TIME_SUB_ADJUST_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#)) 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_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECONDS_CC](#)

2.22.23.2.3 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](#)

2.22.24 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.

2.22.25 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](#)

2.22.25.0.1 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](#)

2.22.25.0.2 Adjusting Time The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE_TIME_SET_TIME_CC](#) or explicitly using [CFE_TIME_SET_STCF_CC](#). TIME provides the ability to command a one time adjustment ([CFE_TIME_ADD_ADJUST_CC](#) and [CFE_TIME_SUB_ADJUST_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#)) 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_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECONDS_CC](#)

2.22.25.0.3 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](#)

2.22.26 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](#)

2.22.27 Adjusting Time

The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE_TIME_SET_TIME_CC](#) or explicitly using [CFE_TIME_SET_STCF_CC](#). TIME provides the ability to command a one time adjustment ([CFE_TIME_ADD_ADJUST_CC](#) and [CFE_TIME_SUB_ADJUST_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#)) 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_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECONDS_CC](#)

2.22.28 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](#)

2.22.29 Frequently Asked Questions about Time Services

None submitted

2.23 cFE Time Services Commands

Upon receipt of any command, the Time Services application will confirm that the message length embedded within the header (from [CFE_MSG_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, TIME will generate the [CFE_TIME_LEN_ERR_EID](#) event, increment the command error counter (\$sc_\$cpu_TIME_CMDEC), and the command will *not* be accepted for processing.

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

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_ADD_ONE_HZ_ADJUSTMENT_CC

Add Delta to Spacecraft Time Correlation Factor each 1Hz

Global CFE_TIME_NOOP_CC

Time No-Op

Global CFE_TIME_RESET_COUNTERS_CC

Time Reset Counters

Global CFE_TIME_SEND_DIAGNOSTIC_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_ADJUST_CC

Subtract Delta from Spacecraft Time Correlation Factor

Global CFE_TIME_SUB_DELAY_CC

Subtract Time from Tone Time Delay

Global CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC

Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

2.24 cFE Time Services Telemetry

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

Global CFE_TIME_DiagnosticTlm_Payload_t

Time Services Diagnostics Packet

Global CFE_TIME_DiagnosticTlm_Payload_t

Time Services Diagnostics Packet

Global CFE_TIME_HousekeepingTlm_Payload_t

Time Services Housekeeping Packet

Global CFE_TIME_HousekeepingTlm_Payload_t

Time Services Housekeeping Packet

2.25 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

Default Time and Tone Order

Global CFE_MISSION_TIME_CFG_DEFAULT_TAI

Default Time Format

Default Time Format

Global CFE_MISSION_TIME_CFG_FAKE_TONE

Default Time Format

Default Time Format

Global CFE_MISSION_TIME_DEF_MET_SECS

Default Time Values

Default Time Values

Global CFE_MISSION_TIME_EPOCH_YEAR

Default EPOCH Values

Default EPOCH Values

Global CFE_MISSION_TIME_FS_FACTOR

Time File System Factor

Time File System Factor

Global CFE_MISSION_TIME_MIN_ELAPSED

Min and Max Time Elapsed

Min and Max Time Elapsed

Global CFE_PLATFORM_TIME_CFG_LATCH_FLY

Define Periodic Time to Update Local Clock Tone Latch

Define Periodic Time to Update Local Clock Tone Latch

Global CFE_PLATFORM_TIME_CFG_SERVER

Time Server or Time Client Selection

Time Server or Time Client Selection

Global CFE_PLATFORM_TIME_CFG_SIGNAL

Include or Exclude the Primary/Redundant Tone Selection Cmd

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

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

Choose the External Time Source for Server only

Global CFE_PLATFORM_TIME_CFG_START_FLY

Define Time to Start Flywheel Since Last Tone

Define Time to Start Flywheel Since Last Tone

Global CFE_PLATFORM_TIME_CFG_TONE_LIMIT

Define Timing Limits From One Tone To The Next

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

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

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

Define the Local Clock Rollover Value in seconds and subseconds

Global CFE_PLATFORM_TIME_START_TASK_PRIORITY

Define TIME Task Priorities

Define TIME Task Priorities

Global CFE_PLATFORM_TIME_START_TASK_STACK_SIZE

Define TIME Task Stack Sizes

Define TIME Task Stack Sizes

2.26 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.

2.27 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_SYS_LOG_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_SYS_LOG_CC
\$sc_\$cpu_ES_OverwriteSysLogMode

Global CFE_ES_QUERY_ALL_CC
\$sc_\$cpu_ES_WriteApplInfo2File

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_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_SYS_LOG_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_EVS_DISABLE_EVENT_TYPE_CC

\$sc_\$cpu_EVS_DisEventType, \$sc_\$cpu_EVS_DisEventTypeMask

Global CFE_EVS_DISABLE_PORTS_CC

\$sc_\$cpu_EVS_DisPort, \$sc_\$cpu_EVS_DisPortMask

Global CFE_EVS_ENABLE_APP_EVENT_TYPE_CC

\$sc_\$cpu_EVS_EnaAppEvtType, \$sc_\$cpu_EVS_EnaAppEvtTypeMask

Global CFE_EVS_ENABLE_APP_EVENTS_CC

\$sc_\$cpu_EVS_EnaAppEvGen

Global CFE_EVS_ENABLE_EVENT_TYPE_CC

\$sc_\$cpu_EVS_EnaEventType, \$sc_\$cpu_EVS_EnaEventTypeMask

Global CFE_EVS_ENABLE_PORTS_CC

\$sc_\$cpu_EVS_EnaPort, \$sc_\$cpu_EVS_EnaPortMask

Global CFE_EVS_NOOP_CC

\$sc_\$cpu_EVS_NOOP

Global CFE_EVS_RESET_ALL_FILTERS_CC

\$sc_\$cpu_EVS_RstAllFltrs

Global CFE_EVS_RESET_APP_COUNTER_CC

\$sc_\$cpu_EVS_RstAppCtrs

Global CFE_EVS_RESET_COUNTERS_CC

\$sc_\$cpu_EVS_ResetCtrs

Global CFE_EVS_RESET_FILTER_CC

\$sc_\$cpu_EVS_RstBinFltrCtr

Global CFE_EVS_SET_EVENT_FORMAT_MODE_CC

\$sc_\$cpu_EVS_SetEvtFmt

Global CFE_EVS_SET_FILTER_CC

\$sc_\$cpu_EVS_SetBinFltrMask

Global CFE_EVS_SET_LOG_MODE_CC

\$sc_\$cpu_EVS_SetLogMode

Global CFE_EVS_WRITE_APP_DATA_FILE_CC

\$sc_\$cpu_EVS_WriteAppData2File

Global CFE_EVS_WRITE_LOG_DATA_FILE_CC

\$sc_\$cpu_EVS_WriteLog2File

Global CFE_SB_DISABLE_ROUTE_CC

\$sc_\$cpu_SB_DisRoute

Global CFE_SB_DISABLE_SUB_REPORTING_CC
\$sc_\$cpu_SB_DisSubRptg

Global CFE_SB_ENABLE_ROUTE_CC
\$sc_\$cpu_SB_EnaRoute

Global CFE_SB_ENABLE_SUB_REPORTING_CC
\$sc_\$cpu_SB_EnaSubRptg

Global CFE_SB_NOOP_CC
\$sc_\$cpu_SB_NOOP

Global CFE_SB_RESET_COUNTERS_CC
\$sc_\$cpu_SB_ResetCtrs

Global CFE_SB_SEND_PREV_SUBS_CC
\$sc_\$cpu_SB_SendPrevSubs

Global CFE_SB_SEND_SB_STATS_CC
\$sc_\$cpu_SB_DumpStats

Global CFE_SB_WRITE_MAP_INFO_CC
\$sc_\$cpu_SB_WriteMap2File

Global CFE_SB_WRITE_PIPE_INFO_CC
\$sc_\$cpu_SB_WritePipe2File

Global CFE_SB_WRITE_ROUTING_INFO_CC
\$sc_\$cpu_SB_WriteRouting2File

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_ADJUST_CC
\$sc_\$cpu_TIME_AddSTCFAdj

```
Global CFE_TIME_ADD_DELAY_CC
$sc_$cpu_TIME_AddClockLat

Global CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC
$sc_$cpu_TIME_Add1HzSTCF

Global CFE_TIME_NOOP_CC
$sc_$cpu_TIME_NOOP

Global CFE_TIME_RESET_COUNTERS_CC
$sc_$cpu_TIME_ResetCtrs

Global CFE_TIME_SEND_DIAGNOSTIC_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_ADJUST_CC
$sc_$cpu_TIME_SubSTCFAdj

Global CFE_TIME_SUB_DELAY_CC
$sc_$cpu_TIME_SubClockLat

Global CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC
$sc_$cpu_TIME_Sub1HzSTCF
```

2.28 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::AddressesAreValid
$sc_$cpu_ES_AddrsValid

Global CFE_ES_AppInfo::BSSAddress
$sc_$cpu_ES_BSSAddress

Global CFE_ES_AppInfo::BSSSize
$sc_$cpu_ES_BSSSize

Global CFE_ES_AppInfo::CodeAddress
$sc_$cpu_ES_CodeAddress
```

Global CFE_ES_AppInfo::CodeSize

\$sc_\$cpu_ES_CodeSize

Global CFE_ES_AppInfo::DataAddress

\$sc_\$cpu_ES_DataAddress

Global CFE_ES_AppInfo::DataSize

\$sc_\$cpu_ES_DataSize

Global CFE_ES_AppInfo::EntryPoint [CFE_MISSION_MAX_API_LEN]

\$sc_\$cpu_ES_AppEntryPt[OS_MAX_API_NAME]

Global CFE_ES_AppInfo::ExceptionAction

\$sc_\$cpu_ES_ExceptnActn

Global CFE_ES_AppInfo::ExecutionCounter

\$sc_\$cpu_ES_ExecutionCtr

Global CFE_ES_AppInfo::FileName [CFE_MISSION_MAX_PATH_LEN]

\$sc_\$cpu_ES_AppFilename[OS_MAX_PATH_LEN]

Global CFE_ES_AppInfo::MainTaskId

\$sc_\$cpu_ES_MainTaskId

Global CFE_ES_AppInfo::MainTaskName [CFE_MISSION_MAX_API_LEN]

\$sc_\$cpu_ES_MainTaskName[OS_MAX_API_NAME]

Global CFE_ES_AppInfo::Name [CFE_MISSION_MAX_API_LEN]

\$sc_\$cpu_ES_AppName[OS_MAX_API_NAME]

Global CFE_ES_AppInfo::NumOfChildTasks

\$sc_\$cpu_ES_ChildTasks

Global CFE_ES_AppInfo::Priority

\$sc_\$cpu_ES_Priority

Global CFE_ES_AppInfo::Resourceld

\$sc_\$cpu_ES_AppID

Global CFE_ES_AppInfo::StackSize

\$sc_\$cpu_ES_StackSize

Global CFE_ES_AppInfo::StartAddress

\$sc_\$cpu_ES_StartAddr

Global CFE_ES_AppInfo::Type

\$sc_\$cpu_ES_AppType

Global CFE_ES_HousekeepingTlm_Payload::BootSource

\$sc_\$cpu_ES_BootSource

Global CFE_ES_HousekeepingTlm_Payload::CFECoreChecksum

\$sc_\$cpu_ES_CKSUM

Global CFE_ES_HousekeepingTlm_Payload::CFEMajorVersion

\$sc_\$cpu_ES_CFEMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::CFEMinorVersion

\$sc_\$cpu_ES_CFE_MINORVER

Global CFE_ES_HousekeepingTlm_Payload::CFEMissionRevision

\$sc_\$cpu_ES_CFEMISSIONREV

```
Global CFE_ES_HousekeepingTlm_Payload::CFERevision
$sc_$cpu_ES_CFEREVISION

Global CFE_ES_HousekeepingTlm_Payload::CommandCounter
$sc_$cpu_ES_CMDPC

Global CFE_ES_HousekeepingTlm_Payload::CommandErrorCounter
$sc_$cpu_ES_CMDEC

Global CFE_ES_HousekeepingTlm_Payload::ERLogEntries
$sc_$cpu_ES_ERLOGENTRIES

Global CFE_ES_HousekeepingTlm_Payload::ERLogIndex
$sc_$cpu_ES_ERLOGINDEX

Global CFE_ES_HousekeepingTlm_Payload::HeapBlocksFree
$sc_$cpu_ES_HeapBlocksFree

Global CFE_ES_HousekeepingTlm_Payload::HeapBytesFree
$sc_$cpu_ES_HeapBytesFree

Global CFE_ES_HousekeepingTlm_Payload::HeapMaxBlockSize
$sc_$cpu_ES_HeapMaxBlkSize

Global CFE_ES_HousekeepingTlm_Payload::MaxProcessorResets
$sc_$cpu_ES_MaxProcResets

Global CFE_ES_HousekeepingTlm_Payload::OSALMajorVersion
$sc_$cpu_ES_OSMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::OSALMinorVersion
$sc_$cpu_ES_OSMINORVER

Global CFE_ES_HousekeepingTlm_Payload::OSALMissionRevision
$sc_$cpu_ES_OSMISSIONREV

Global CFE_ES_HousekeepingTlm_Payload::OSALRevision
$sc_$cpu_ES_OSREVISION

Global CFE_ES_HousekeepingTlm_Payload::PerfDataCount
$sc_$cpu_ES_PerfDataCnt

Global CFE_ES_HousekeepingTlm_Payload::PerfDataEnd
$sc_$cpu_ES_PerfDataEnd

Global CFE_ES_HousekeepingTlm_Payload::PerfDataStart
$sc_$cpu_ES_PerfDataStart

Global CFE_ES_HousekeepingTlm_Payload::PerfData2Write
$sc_$cpu_ES_PerfData2Write

Global CFE_ES_HousekeepingTlm_Payload::PerfFilterMask [CFE_MISSION_ES_PERF_MAX_IDS/32]
$sc_$cpu_ES_PerfFltrMask[MaskCnt]

Global CFE_ES_HousekeepingTlm_Payload::PerfMode
$sc_$cpu_ES_PerfMode

Global CFE_ES_HousekeepingTlm_Payload::PerfState
$sc_$cpu_ES_PerfState

Global CFE_ES_HousekeepingTlm_Payload::PerfTriggerCount
$sc_$cpu_ES_PerfTrigCnt
```

Global CFE_ES_HousekeepingTlm_Payload::PerfTriggerMask [CFE_MISSION_ES_PERF_MAX_IDS/32]
\$sc_\$cpu_ES_PerfTrigMask[MaskCnt]

Global CFE_ES_HousekeepingTlm_Payload::ProcessorResets
\$sc_\$cpu_ES_ProcResetCnt

Global CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion
\$sc_\$cpu_ES_PSPMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion
\$sc_\$cpu_ES_PSPMINORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision
\$sc_\$cpu_ES_PSPMISSIONREV

Global CFE_ES_HousekeepingTlm_Payload::PSPRevision
\$sc_\$cpu_ES_PSPREVISION

Global CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps
\$sc_\$cpu_ES_RegCoreApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps
\$sc_\$cpu_ES_RegExtApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredLibs
\$sc_\$cpu_ES_RegLibs

Global CFE_ES_HousekeepingTlm_Payload::RegisteredTasks
\$sc_\$cpu_ES_RegTasks

Global CFE_ES_HousekeepingTlm_Payload::ResetSubtype
\$sc_\$cpu_ES_ResetSubtype

Global CFE_ES_HousekeepingTlm_Payload::ResetType
\$sc_\$cpu_ES_ResetType

Global CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed
\$sc_\$cpu_ES_SYSLOGBYTEUSED

Global CFE_ES_HousekeepingTlm_Payload::SysLogEntries
\$sc_\$cpu_ES_SYSLOGENTRIES

Global CFE_ES_HousekeepingTlm_Payload::SysLogMode
\$sc_\$cpu_ES_SYSLOGMODE

Global CFE_ES_HousekeepingTlm_Payload::SysLogSize
\$sc_\$cpu_ES_SYSLOGSIZE

Global CFE_ES_MemPoolStats::BlockStats [CFE_MISSION_ES_POOL_MAX_BUCKETS]
\$sc_\$cpu_ES_BlkStats[BLK_SIZES]

Global CFE_ES_MemPoolStats::CheckErrCtr
\$sc_\$cpu_ES_BlkErrCTR

Global CFE_ES_MemPoolStats::NumBlocksRequested
\$sc_\$cpu_ES_BlkREQ

Global CFE_ES_MemPoolStats::NumFreeBytes
\$sc_\$cpu_ES_FreeBytes

Global CFE_ES_MemPoolStats::PoolSize
\$sc_\$cpu_ES_PoolSize

Global CFE_ES_PoolStatsTlm_Payload::PoolHandle
\$sc_\$cpu_ES_PoolHandle

Global CFE_EVS_AppTlmData::AppEnableStatus
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPENABLESTAT

Global CFE_EVS_AppTlmData::AppID
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPID

Global CFE_EVS_AppTlmData::AppMessageSentCounter
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPMGSENTC

Global CFE_EVS_AppTlmData::AppMessageSquelchedCounter
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].SQUELCHEDC

Global CFE_EVS_HousekeepingTlm_Payload::AppData [CFE_MISSION_ES_MAX_APPLICATIONS]
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS]

Global CFE_EVS_HousekeepingTlm_Payload::CommandCounter
\$sc_\$cpu_EVS_CMDPC

Global CFE_EVS_HousekeepingTlm_Payload::CommandErrorCounter
\$sc_\$cpu_EVS_CMDEC

Global CFE_EVS_HousekeepingTlm_Payload::LogEnabled
\$sc_\$cpu_EVS_LOGENABLED

Global CFE_EVS_HousekeepingTlm_Payload::LogFullFlag
\$sc_\$cpu_EVS_LOGFULL

Global CFE_EVS_HousekeepingTlm_Payload::LogMode
\$sc_\$cpu_EVS_LOGMODE

Global CFE_EVS_HousekeepingTlm_Payload::LogOverflowCounter
\$sc_\$cpu_EVS_LOGOVERFLOWC

Global CFE_EVS_HousekeepingTlm_Payload::MessageFormatMode
\$sc_\$cpu_EVS_MSGFMTMODE

Global CFE_EVS_HousekeepingTlm_Payload::MessageSendCounter
\$sc_\$cpu_EVS_MSGSENTC

Global CFE_EVS_HousekeepingTlm_Payload::MessageTruncCounter
\$sc_\$cpu_EVS_MSGTRUNC

Global CFE_EVS_HousekeepingTlm_Payload::OutputPort
\$sc_\$cpu_EVS_OUTPUTPORT

Global CFE_EVS_HousekeepingTlm_Payload::Spare1
\$sc_\$cpu_EVS_HK_SPARE1

Global CFE_EVS_HousekeepingTlm_Payload::Spare2
\$sc_\$cpu_EVS_HK_SPARE2

Global CFE_EVS_HousekeepingTlm_Payload::Spare3
\$sc_\$cpu_EVS_HK_SPARE3

Global CFE_EVS_HousekeepingTlm_Payload::UnregisteredAppCounter
\$sc_\$cpu_EVS_UNREGAPPC

Global CFE_EVS_LongEventTlm_Payload::Message [CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]
\$sc_\$cpu_EVENT[CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]

Global CFE_EVS_LongEventTlm_Payload::Spare1
\$sc_\$cpu_EVS_SPARE1

Global CFE_EVS_LongEventTlm_Payload::Spare2
\$sc_\$cpu_EVS_SPARE2

Global CFE_EVS_PacketID::AppName [CFE_MISSION_MAX_API_LEN]
\$sc_\$cpu_EVS_APPNAME[OS_MAX_API_NAME]

Global CFE_EVS_PacketID::EventID
\$sc_\$cpu_EVS_EVENTID

Global CFE_EVS_PacketID::EventType
\$sc_\$cpu_EVS_EVENTTYPE

Global CFE_EVS_PacketID::ProcessorID
\$sc_\$cpu_EVS_PROCESSORID

Global CFE_EVS_PacketID::SpacecraftID
\$sc_\$cpu_EVS_SCID

Global CFE_SB_HousekeepingTlm_Payload::CommandCounter
\$sc_\$cpu_SB_CMDPC

Global CFE_SB_HousekeepingTlm_Payload::CommandErrorCounter
\$sc_\$cpu_SB_CMDEC

Global CFE_SB_HousekeepingTlm_Payload::CreatePipeErrorCounter
\$sc_\$cpu_SB_NewPipeEC

Global CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptionsCounter
\$sc_\$cpu_SB_DupSubCnt

Global CFE_SB_HousekeepingTlm_Payload::GetPipeldByNameErrorCounter
\$sc_\$cpu_SB_GetPipeIDByNameEC

Global CFE_SB_HousekeepingTlm_Payload::InternalErrorCounter
\$sc_\$cpu_SB_InternalEC

Global CFE_SB_HousekeepingTlm_Payload::MemInUse
\$sc_\$cpu_SB_MemInUse

Global CFE_SB_HousekeepingTlm_Payload::MemPoolHandle
\$sc_\$cpu_SB_MemPoolHdl

Global CFE_SB_HousekeepingTlm_Payload::MsgLimitErrorCounter
\$sc_\$cpu_SB_MsgLimEC

Global CFE_SB_HousekeepingTlm_Payload::MsgReceiveErrorCounter
\$sc_\$cpu_SB_MsgRecEC

Global CFE_SB_HousekeepingTlm_Payload::MsgSendErrorCounter
\$sc_\$cpu_SB_MsgSndEC

Global CFE_SB_HousekeepingTlm_Payload::NoSubscribersCounter
\$sc_\$cpu_SB_NoSubEC

Global CFE_SB_HousekeepingTlm_Payload::PipeOptsErrorCounter
\$sc_\$cpu_SB_PipeOptsEC

Global CFE_SB_HousekeepingTlm_Payload::PipeOverflowErrorCounter
\$sc_\$cpu_SB_PipeOvrEC

```
Global CFE_SB_HousekeepingTlm_Payload::Spare2Align [1]
$sc_$cpu_SB_Spare2Align[2]

Global CFE_SB_HousekeepingTlm_Payload::SubscribeErrorCounter
$sc_$cpu_SB_SubscrEC

Global CFE_SB_HousekeepingTlm_Payload::UnmarkedMem
$sc_$cpu_SB_UnMarkedMem

Global CFE_SB_PipeDepthStats::CurrentQueueDepth
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDINUSE

Global CFE_SB_PipeDepthStats::MaxQueueDepth
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDDEPTH

Global CFE_SB_PipeDepthStats::PeakQueueDepth
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDPKINUSE

Global CFE_SB_PipeDepthStats::PipeId
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDPIPEID

Global CFE_SB_PipeDepthStats::Spare
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDSPARE

Global CFE_SB_StatsTlm_Payload::MaxMemAllowed
$sc_$cpu_SB_Stat.SB_SMMBMALW

Global CFE_SB_StatsTlm_Payload::MaxMsgIdsAllowed
$sc_$cpu_SB_Stat.SB_SMMMDALW

Global CFE_SB_StatsTlm_Payload::MaxPipeDepthAllowed
$sc_$cpu_SB_Stat.SB_SMMPDALW

Global CFE_SB_StatsTlm_Payload::MaxPipesAllowed
$sc_$cpu_SB_Stat.SB_SMMPALW

Global CFE_SB_StatsTlm_Payload::MaxSubscriptionsAllowed
$sc_$cpu_SB_Stat.SB_SMMSALW

Global CFE_SB_StatsTlm_Payload::MemInUse
$sc_$cpu_SB_Stat.SB_SMBMIU

Global CFE_SB_StatsTlm_Payload::MsgIdsInUse
$sc_$cpu_SB_Stat.SB_SMMIDIU

Global CFE_SB_StatsTlm_Payload::PeakMemInUse
$sc_$cpu_SB_Stat.SB_SMPBMIU

Global CFE_SB_StatsTlm_Payload::PeakMsgIdsInUse
$sc_$cpu_SB_Stat.SB_SMPMIDIU

Global CFE_SB_StatsTlm_Payload::PeakPipesInUse
$sc_$cpu_SB_Stat.SB_SMPPIU

Global CFE_SB_StatsTlm_Payload::PeakSBBuffersInUse
$sc_$cpu_SB_Stat.SB_SMPSSBIU

Global CFE_SB_StatsTlm_Payload::PeakSubscriptionsInUse
$sc_$cpu_SB_Stat.SB_SMPSIU

Global CFE_SB_StatsTlm_Payload::PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]
$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES]
```

Global CFE_SB_StatsTlm_Payload::PipesInUse
\$sc_\$cpu_SB_Stat.SB_SMPIU

Global CFE_SB_StatsTlm_Payload::SBBuffersInUse
\$sc_\$cpu_SB_Stat.SB_SMSBBIU

Global CFE_SB_StatsTlm_Payload::SubscriptionsInUse
\$sc_\$cpu_SB_Stat.SB_SMSIU

Global CFE_TBL_HousekeepingTlm_Payload::ActiveBuffer
\$sc_\$cpu_TBL_LastValBuf

Global CFE_TBL_HousekeepingTlm_Payload::ByteAlignPad1
\$sc_\$cpu_TBL_BytAlignPad1

Global CFE_TBL_HousekeepingTlm_Payload::CommandCounter
\$sc_\$cpu_TBL_CMDPC

Global CFE_TBL_HousekeepingTlm_Payload::CommandErrorCounter
\$sc_\$cpu_TBL_CMDEC

Global CFE_TBL_HousekeepingTlm_Payload::FailedValCounter
\$sc_\$cpu_TBL_ValFailedCtr

Global CFE_TBL_HousekeepingTlm_Payload::LastFileDumped [CFE_MISSION_MAX_PATH_LEN]
\$sc_\$cpu_TBL_LastFileDumped[OS_MAX_PATH_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]
\$sc_\$cpu_TBL_LastFileLoaded[OS_MAX_PATH_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::LastTableLoaded [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
\$sc_\$cpu_TBL_LastTableLoaded[CFE_TBL_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_L←EN]
\$sc_\$cpu_TBL_LastUpdTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::LastUpdateTime
\$sc_\$cpu_TBL_LastUpdTm, \$sc_\$cpu_TBL_SECONDS, \$sc_\$cpu_TBL_SUBSECONDS

Global CFE_TBL_HousekeepingTlm_Payload::LastValCrc
\$sc_\$cpu_TBL_LastValCRC

Global CFE_TBL_HousekeepingTlm_Payload::LastValStatus
\$sc_\$cpu_TBL_LastVals

Global CFE_TBL_HousekeepingTlm_Payload::LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_L←EN]
\$sc_\$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::MemPoolHandle
\$sc_\$cpu_TBL_MemPoolHandle

Global CFE_TBL_HousekeepingTlm_Payload::NumFreeSharedBufs
\$sc_\$cpu_TBL_NumFreeShrBuf

Global CFE_TBL_HousekeepingTlm_Payload::NumLoadPending
\$sc_\$cpu_TBL_NumUpdatesPend

Global CFE_TBL_HousekeepingTlm_Payload::NumTables
\$sc_\$cpu_TBL_NumTables

Global **CFE_TBL_HousekeepingTlm_Payload::NumValRequests**
\$sc_\$cpu_TBL_ValReqCtr

Global **CFE_TBL_HousekeepingTlm_Payload::SuccessValCounter**
\$sc_\$cpu_TBL_ValSuccessCtr

Global **CFE_TBL_HousekeepingTlm_Payload::ValidationCounter**
\$sc_\$cpu_TBL_ValCompltdCtr

Global **CFE_TBL_TblRegPacket_Payload::ActiveBufferAddr**
\$sc_\$cpu_TBL_ActBufAdd

Global **CFE_TBL_TblRegPacket_Payload::ByteAlign4**
\$sc_\$cpu_TBL_Spare4

Global **CFE_TBL_TblRegPacket_Payload::Crc**
\$sc_\$cpu_TBL_CRC

Global **CFE_TBL_TblRegPacket_Payload::Critical**
\$sc_\$cpu_TBL_Spare3

Global **CFE_TBL_TblRegPacket_Payload::DoubleBuffered**
\$sc_\$cpu_TBL_DblBuffered

Global **CFE_TBL_TblRegPacket_Payload::DumpOnly**
\$sc_\$cpu_TBL_DumpOnly

Global **CFE_TBL_TblRegPacket_Payload::FileTime**
\$sc_\$cpu_TBL_FILETIME

Global **CFE_TBL_TblRegPacket_Payload::InactiveBufferAddr**
\$sc_\$cpu_TBL_IActBufAdd

Global **CFE_TBL_TblRegPacket_Payload::LastFileLoaded** [CFE_MISSION_MAX_PATH_LEN]
\$sc_\$cpu_TBL_LastFileUpd[OS_MAX_PATH_LEN]

Global **CFE_TBL_TblRegPacket_Payload::LoadPending**
\$sc_\$cpu_TBL_UpdatePndng

Global **CFE_TBL_TblRegPacket_Payload::Name** [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
\$sc_\$cpu_TBL_Name[CFE_TB_MAX_FULL_NAME_LEN]

Global **CFE_TBL_TblRegPacket_Payload::OwnerAppName** [CFE_MISSION_MAX_API_LEN]
\$sc_\$cpu_TBL_OwnerApp[OS_MAX_API_NAME]

Global **CFE_TBL_TblRegPacket_Payload::Size**
\$sc_\$cpu_TBL_SIZE

Global **CFE_TBL_TblRegPacket_Payload::TableLoadedOnce**
\$sc_\$cpu_TBL_LoadedOnce

Global **CFE_TBL_TblRegPacket_Payload::TimeOfLastUpdate**
\$sc_\$cpu_TBL_TimeLastUpd, \$sc_\$cpu_TBL_TLUSECONDS, \$sc_\$cpu_TBL_TLUSUBSECONDS

Global **CFE_TBL_TblRegPacket_Payload::ValidationFuncPtr**
\$sc_\$cpu_TBL_ValFuncPtr

Global **CFE_TIME_DiagnosticTlm_Payload::AtToneDelay**
\$sc_\$cpu_TIME_DLlatentS, \$sc_\$cpu_TIME_DLlatentSs

Global **CFE_TIME_DiagnosticTlm_Payload::AtToneLatch**
\$sc_\$cpu_TIME_DTVvalidS, \$sc_\$cpu_TIME_DTVvalidSs

Global CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds
\$sc_\$cpu_TIME_DLearS

Global CFE_TIME_DiagnosticTlm_Payload::AtToneMET
\$sc_\$cpu_TIME_DMETS, \$sc_\$cpu_TIME_DMETSS

Global CFE_TIME_DiagnosticTlm_Payload::AtToneSTCF
\$sc_\$cpu_TIME_DSTCFS, \$sc_\$cpu_TIME_DSTCFSS

Global CFE_TIME_DiagnosticTlm_Payload::ClockFlyState
\$sc_\$cpu_TIME_DFlywheel

Global CFE_TIME_DiagnosticTlm_Payload::ClockSetState
\$sc_\$cpu_TIME_DValid

Global CFE_TIME_DiagnosticTlm_Payload::ClockSignal
\$sc_\$cpu_TIME_DSignal

Global CFE_TIME_DiagnosticTlm_Payload::ClockSource
\$sc_\$cpu_TIME_DSource

Global CFE_TIME_DiagnosticTlm_Payload::ClockStateAPI
\$sc_\$cpu_TIME_DAPIS

Global CFE_TIME_DiagnosticTlm_Payload::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_DFlag←
NIU

Global CFE_TIME_DiagnosticTlm_Payload::CurrentLatch
\$sc_\$cpu_TIME_DLlocalS, \$sc_\$cpu_TIME_DLlocalSs

Global CFE_TIME_DiagnosticTlm_Payload::CurrentMET
\$sc_\$cpu_TIME_DMETS, \$sc_\$cpu_TIME_DMETSS

Global CFE_TIME_DiagnosticTlm_Payload::CurrentTAI
\$sc_\$cpu_TIME_DTAIS, \$sc_\$cpu_TIME_DTAISS

Global CFE_TIME_DiagnosticTlm_Payload::CurrentUTC
\$sc_\$cpu_TIME_DUTCS, \$sc_\$cpu_TIME_DUTCSS

Global CFE_TIME_DiagnosticTlm_Payload::DataStoreStatus
\$sc_\$cpu_TIME_DataStStat

Global CFE_TIME_DiagnosticTlm_Payload::DelayDirection
\$sc_\$cpu_TIME_DLlatentDir

Global CFE_TIME_DiagnosticTlm_Payload::Forced2Fly
\$sc_\$cpu_TIME_DCMD2Fly

Global CFE_TIME_DiagnosticTlm_Payload::LocalIntCounter
\$sc_\$cpu_TIME_D1HzSRCNT

Global CFE_TIME_DiagnosticTlm_Payload::LocalTaskCounter
\$sc_\$cpu_TIME_D1HzTaskCNT

Global CFE_TIME_DiagnosticTlm_Payload::MaxElapsed
\$sc_\$cpu_TIME_DMaxWindow

Global CFE_TIME_DiagnosticTlm_Payload::MaxLocalClock
\$sc_\$cpu_TIME_DWrapS, \$sc_\$cpu_TIME_DWrapSs

Global CFE_TIME_DiagnosticTlm_Payload::MinElapsed
\$sc_\$cpu_TIME_DMinWindow

Global CFE_TIME_DiagnosticTlm_Payload::OneHzAdjust
\$sc_\$cpu_TIME_D1HzAdjS, \$sc_\$cpu_TIME_D1HzAdjSs

Global CFE_TIME_DiagnosticTlm_Payload::OneHzDirection
\$sc_\$cpu_TIME_D1HzAdjDir

Global CFE_TIME_DiagnosticTlm_Payload::OneTimeAdjust
\$sc_\$cpu_TIME_DAdjustS, \$sc_\$cpu_TIME_DAdjustSs

Global CFE_TIME_DiagnosticTlm_Payload::OneTimeDirection
\$sc_\$cpu_TIME_DAdjustDir

Global CFE_TIME_DiagnosticTlm_Payload::ServerFlyState
\$sc_\$cpu_TIME_DSrvFly

Global CFE_TIME_DiagnosticTlm_Payload::TimeSinceTone
\$sc_\$cpu_TIME_DElapsedS, \$sc_\$cpu_TIME_DElapsedSs

Global CFE_TIME_DiagnosticTlm_Payload::ToneDataCounter
\$sc_\$cpu_TIME_DTatTCNT

Global CFE_TIME_DiagnosticTlm_Payload::ToneDataLatch
\$sc_\$cpu_TIME_DTDS, \$sc_\$cpu_TIME_DTDSSs

Global CFE_TIME_DiagnosticTlm_Payload::ToneIntCounter
\$sc_\$cpu_TIME_DTsISRCNT

Global CFE_TIME_DiagnosticTlm_Payload::ToneIntErrorCounter
\$sc_\$cpu_TIME_DTsISRERR

Global CFE_TIME_DiagnosticTlm_Payload::ToneMatchCounter
\$sc_\$cpu_TIME_DVerifyCNT

Global CFE_TIME_DiagnosticTlm_Payload::ToneMatchErrorCounter
\$sc_\$cpu_TIME_DVerifyER

Global CFE_TIME_DiagnosticTlm_Payload::ToneOverLimit
\$sc_\$cpu_TIME_DMaxSs

Global CFE_TIME_DiagnosticTlm_Payload::ToneSignalCounter
\$sc_\$cpu_TIME_DTSDetCNT

Global CFE_TIME_DiagnosticTlm_Payload::ToneSignalLatch
\$sc_\$cpu_TIME_DTTS, \$sc_\$cpu_TIME_DTTSSs

Global CFE_TIME_DiagnosticTlm_Payload::ToneTaskCounter
\$sc_\$cpu_TIME_DTsTaskCNT

Global CFE_TIME_DiagnosticTlm_Payload::ToneUnderLimit
\$sc_\$cpu_TIME_DMinSs

Global CFE_TIME_DiagnosticTlm_Payload::VersionCounter
\$sc_\$cpu_TIME_DVersionCNT

Global CFE_TIME_DiagnosticTlm_Payload::VirtualMET
\$sc_\$cpu_TIME_DLLogicalMET

Global CFE_TIME_HousekeepingTlm_Payload::ClockStateAPI

\$sc_\$cpu_TIME_DAPIState

Global CFE_TIME_HousekeepingTlm_Payload::ClockStateFlags

\$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

Global CFE_TIME_HousekeepingTlm_Payload::CommandCounter

\$sc_\$cpu_TIME_CMDPC

Global CFE_TIME_HousekeepingTlm_Payload::CommandErrorCounter

\$sc_\$cpu_TIME_CMDEC

Global CFE_TIME_HousekeepingTlm_Payload::LeapSeconds

\$sc_\$cpu_TIME_LeapSecs

Global CFE_TIME_HousekeepingTlm_Payload::Seconds1HzAdj

\$sc_\$cpu_TIME_1HzAdjSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsDelay

\$sc_\$cpu_TIME_1HzAdjSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsMET

\$sc_\$cpu_TIME_METSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsSTCF

\$sc_\$cpu_TIME_STCFSecs

Global CFE_TIME_HousekeepingTlm_Payload::Subsecs1HzAdj

\$sc_\$cpu_TIME_1HzAdjSSecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsDelay

\$sc_\$cpu_TIME_1HzAdjSSecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsMET

\$sc_\$cpu_TIME_METSubsecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsSTCF

\$sc_\$cpu_TIME_STCFSubsecs

3 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
Platform Support Package (PSP)	A collection of user-provided facilities that interface an OS and the cFE with a specific hardware platform. The PSP is responsible for hardware initialization.
Child Task	A separate thread of execution that is spawned by an Application's Main Task.

Term	Definition
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_EventType_DEBUG , CFE_EVS_EventType_INFORMATION , CFE_EVS_EventType_ERROR , or CFE_EVS_EventType_CRITICAL .
Event Message	A data item used to notify the user and/or an external Application of a significant event. Event Messages include a time-stamp 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.

4 cFE Application Programmer's Interface (API) Reference

4.1 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.
- [cFE Application Control APIs](#)
 - [CFE_ES_RestartApp](#) - Restart a single cFE Application.
 - [CFE_ES_ReloadApp](#) - Reload a single cFE Application.
 - [CFE_ES_DeleteApp](#) - Delete a cFE Application.
- [cFE Application Behavior APIs](#)
 - [CFE_ES_RunLoop](#) - Check for Exit, Restart, or Reload commands.
 - [CFE_ES_WaitForStartupSync](#) - Allow an Application to Wait for the "OPERATIONAL" global system state.
 - [CFE_ES_WaitForSystemState](#) - Allow an Application to Wait for a minimum global system state.

- [CFE_ES_IncrementTaskCounter](#) - Increments the execution counter for the calling task.
- [CFE_ES_ExitApp](#) - Exit a cFE Application.
- [cFE Information APIs](#)
 - [CFE_ES_GetResetType](#) - Return the most recent Reset Type.
 - [CFE_ES_GetAppID](#) - Get an Application ID for the calling Application.
 - [CFE_ES_GetTaskID](#) - Get the task ID of the calling context.
 - [CFE_ES_GetAppIDByName](#) - Get an Application ID associated with a specified Application name.
 - [CFE_ES_GetLibIDByName](#) - Get a Library ID associated with a specified Library name.
 - [CFE_ES_GetAppName](#) - Get an Application name for a specified Application ID.
 - [CFE_ES_GetLibName](#) - Get a Library name for a specified Library ID.
 - [CFE_ES_GetAppInfo](#) - Get Application Information given a specified App ID.
 - [CFE_ES_GetTaskInfo](#) - Get Task Information given a specified Task ID.
 - [CFE_ES_GetLibInfo](#) - Get Library Information given a specified Resource ID.
 - [CFE_ES_GetModuleInfo](#) - Get Information given a specified Resource ID.
- [cFE Child Task APIs](#)
 - [CFE_ES_CreateChildTask](#) - Creates a new task under an existing Application.
 - [CFE_ES_GetTaskIDByName](#) - Get a Task ID associated with a specified Task name.
 - [CFE_ES_GetTaskName](#) - Get a Task name for a specified Task ID.
 - [CFE_ES_DeleteChildTask](#) - Deletes a task under an existing Application.
 - [CFE_ES_ExitChildTask](#) - Exits a child task.
- [cFE Critical Data Store APIs](#)
 - [CFE_ES_RegisterCDS](#) - Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
 - [CFE_ES_GetCDSBlockIDByName](#) - Get a CDS Block ID associated with a specified CDS Block name.
 - [CFE_ES_GetCDSBlockName](#) - Get a Block name for a specified Block ID.
 - [CFE_ES_CopyToCDS](#) - Save a block of data in the Critical Data Store (CDS)
 - [CFE_ES_RestoreFromCDS](#) - Recover a block of data from the Critical Data Store (CDS)
- [cFE Memory Manager APIs](#)
 - [CFE_ES_PoolCreate](#) - Initializes a memory pool created by an application while using a semaphore during processing.
 - [CFE_ES_PoolCreateEx](#) - Initializes a memory pool created by an application with application specified block sizes.
 - [CFE_ES_PoolCreateNoSem](#) - Initializes a memory pool created by an application without using a semaphore during processing.
 - [CFE_ES_PoolDelete](#) - Deletes a memory pool that was previously created.
 - [CFE_ES_GetPoolBuf](#) - Gets a buffer from the memory pool created by [CFE_ES_PoolCreate](#) or [CFE_ES_PoolCreateNoSem](#).
 - [CFE_ES_PutPoolBuf](#) - Releases a buffer from the memory pool that was previously allocated via [CFE_ES_GetPoolBuf](#).
 - [CFE_ES_GetMemPoolStats](#) - Extracts the statistics maintained by the memory pool software.
 - [CFE_ES_GetPoolBufInfo](#) - Gets info on a buffer previously allocated via [CFE_ES_GetPoolBuf](#).

- cFE Performance Monitor APIs
 - [CFE_ES_PerfLogEntry](#) - Entry marker for use with Software Performance Analysis Tool.
 - [CFE_ES_PerfLogExit](#) - Exit marker for use with Software Performance Analysis Tool.
 - [CFE_ES_PerfLogAdd](#) - Adds a new entry to the data buffer.
- cFE Generic Counter APIs
 - [CFE_ES_RegisterGenCounter](#) - Register a generic counter.
 - [CFE_ES_DeleteGenCounter](#) - Delete a generic counter.
 - [CFE_ES_IncrementGenCounter](#) - Increments the specified generic counter.
 - [CFE_ES_SetGenCount](#) - Set the specified generic counter.
 - [CFE_ES_GetGenCount](#) - Get the specified generic counter count.
 - [CFE_ES_GetGenCounterIDByName](#) - Get the Id associated with a generic counter name.
 - [CFE_ES_GetGenCounterName](#) - Get a Counter name for a specified Counter ID.
- cFE Miscellaneous APIs
 - [CFE_ES_BackgroundWakeup](#) - Wakes up the CFE background task.
 - [CFE_ES_CalculateCRC](#) - Calculate a CRC on a block of memory.
 - [CFE_ES_WriteToSysLog](#) - Write a string to the cFE System Log.
 - [CFE_ES_ProcessAsyncEvent](#) - Notification that an asynchronous event was detected by the underlying OS/PSP.
 - [CFE_ES_StatusToString](#) - Convert status to a string.
- cFE Resource ID APIs
 - [CFE_ES_AppID_ToIndex](#) - Obtain an index value correlating to an ES Application ID.
 - [CFE_ES_LibID_ToIndex](#) - Obtain an index value correlating to an ES Library ID.
 - [CFE_ES_TaskID_ToIndex](#) - Obtain an index value correlating to an ES Task ID.
 - [CFE_ES_CounterID_ToIndex](#) - Obtain an index value correlating to an ES Counter ID.

4.2 Events Services API

- cFE Registration APIs
 - [CFE_EVS_Register](#) - Register an application for receiving event services.
- cFE Send Event APIs
 - [CFE_EVS_SendEvent](#) - Generate a software event.
 - [CFE_EVS_SendEventWithAppID](#) - Generate a software event given the specified Application ID.
 - [CFE_EVS_SendTimedEvent](#) - Generate a software event with a specific time tag.
- cFE Reset Event Filter APIs
 - [CFE_EVS_ResetFilter](#) - Resets the calling application's event filter for a single event ID.
 - [CFE_EVS_ResetAllFilters](#) - Resets all of the calling application's event filters.

4.3 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 specified Standard cFE File Header to the specified file.
 - [CFE_FS_SetTimestamp](#) - Modifies the Time Stamp field in the Standard cFE File Header for the specified file.
- cFE File Utility APIs
 - [CFE_FS_GetDefaultMountPoint](#) - Get the default virtual mount point for a file category.
 - [CFE_FS_GetDefaultExtension](#) - Get the default filename extension for a file category.
 - [CFE_FS_ParseInputFileNameEx](#) - Parse a filename input from an input buffer into a local buffer.
 - [CFE_FS_ParseInputFileName](#) - Parse a filename string from the user into a local buffer.
 - [CFE_FS_ExtractFilenameFromPath](#) - Extracts the filename from a unix style path and filename string.
 - [CFE_FS_BackgroundFileDumpRequest](#) - Register a background file dump request.
 - [CFE_FS_BackgroundFileDumpsPending](#) - Query if a background file write request is currently pending.

4.4 Message API

- cFE Generic Message APIs
 - [CFE_MSG_Init](#) - Initialize a message.
- cFE Message Primary Header APIs
 - [CFE_MSG.GetSize](#) - Gets the total size of a message.
 - [CFE_MSG_SetSize](#) - Sets the total size of a message.
 - [CFE_MSG.GetType](#) - Gets the message type.
 - [CFE_MSG_SetType](#) - Sets the message type.
 - [CFE_MSG_GetHeaderVersion](#) - Gets the message header version.
 - [CFE_MSG_SetHeaderVersion](#) - Sets the message header version.
 - [CFE_MSG_GetHasSecondaryHeader](#) - Gets the message secondary header boolean.
 - [CFE_MSG_SetHasSecondaryHeader](#) - Sets the message secondary header boolean.
 - [CFE_MSG_GetApId](#) - Gets the message application ID.
 - [CFE_MSG_SetApId](#) - Sets the message application ID.
 - [CFE_MSG_GetSegmentationFlag](#) - Gets the message segmentation flag.
 - [CFE_MSG_SetSegmentationFlag](#) - Sets the message segmentation flag.
 - [CFE_MSG_GetSequenceCount](#) - Gets the message sequence count.
 - [CFE_MSG_SetSequenceCount](#) - Sets the message sequence count.
 - [CFE_MSG_GetNextSequenceCount](#) - Gets the next sequence count value (rolls over if appropriate)
- cFE Message Extended Header APIs
 - [CFE_MSG_GetEDSVersion](#) - Gets the message EDS version.
 - [CFE_MSG_SetEDSVersion](#) - Sets the message EDS version.
 - [CFE_MSG_GetEndian](#) - Gets the message endian.

- [CFE_MSG_SetEndian](#) - Sets the message endian.
 - [CFE_MSG_GetPlaybackFlag](#) - Gets the message playback flag.
 - [CFE_MSG_SetPlaybackFlag](#) - Sets the message playback flag.
 - [CFE_MSG_GetSubsystem](#) - Gets the message subsystem.
 - [CFE_MSG_SetSubsystem](#) - Sets the message subsystem.
 - [CFE_MSG_GetSystem](#) - Gets the message system.
 - [CFE_MSG_SetSystem](#) - Sets the message system.
- cFE Message Secondary Header APIs
 - [CFE_MSG_GenerateChecksum](#) - Calculates and sets the checksum of a message.
 - [CFE_MSG_ValidateChecksum](#) - Validates the checksum of a message.
 - [CFE_MSG_SetFcnCode](#) - Sets the function code field in a message.
 - [CFE_MSG_GetFcnCode](#) - Gets the function code field from a message.
 - [CFE_MSG_GetMsgTime](#) - Gets the time field from a message.
 - [CFE_MSG_SetMsgTime](#) - Sets the time field in a message.
 - cFE Message Id APIs
 - [CFE_MSG_GetMsgId](#) - Gets the message id from a message.
 - [CFE_MSG_SetMsgId](#) - Sets the message id bits in a message.
 - [CFE_MSG.GetTypeFromMsgId](#) - Gets message type using message ID.

4.5 Resource ID API

- cFE Resource Misc APIs
 - [CFE_Resourceld_ToInteger](#) - Convert a resource ID to an integer.
 - [CFE_Resourceld_FromInteger](#) - Convert an integer to a resource ID.
 - [CFE_Resourceld_Equal](#) - Compare two Resource ID values for equality.
 - [CFE_Resourceld_IsDefined](#) - Check if a resource ID value is defined.
 - [CFE_Resourceld_GetBase](#) - Get the Base value (type/category) from a resource ID value.
 - [CFE_Resourceld_GetSerial](#) - Get the Serial Number (sequential ID) from a resource ID value.
 - [CFE_Resourceld_FindNext](#) - Locate the next resource ID which does not map to an in-use table entry.
 - [CFE_Resourceld_ToIndex](#) - Internal routine to aid in converting an ES resource ID to an array index.

4.6 Software Bus Services API

- cFE Pipe Management APIs
 - [CFE_SB_CreatePipe](#) - Creates a new software bus pipe.
 - [CFE_SB_DeletePipe](#) - Delete a software bus pipe.
 - [CFE_SB_Pipeld_ToIndex](#) - Obtain an index value correlating to an SB Pipe ID.
 - [CFE_SB_SetPipeOpts](#) - Set options on a pipe.
 - [CFE_SB_GetPipeOpts](#) - Get options on a pipe.
 - [CFE_SB_GetPipeName](#) - Get the pipe name for a given id.
 - [CFE_SB_GetPipeldByName](#) - Get pipe id by pipe name.
- cFE 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 local to a cpu.
- [CFE_SB_Unsubscribe](#) - Remove a subscription to a message on the software bus.
- [CFE_SB_UnsubscribeLocal](#) - Remove a subscription to a message on the software bus on the current CPU.
- [cFE Send/Receive Message APIs](#)
 - [CFE_SB_TransmitMsg](#) - Transmit a message.
 - [CFE_SB_ReceiveBuffer](#) - Receive a message from a software bus pipe.
- [cFE Zero Copy APIs](#)
 - [CFE_SB_AllocateMessageBuffer](#) - Get a buffer pointer to use for "zero copy" SB sends.
 - [CFE_SB_ReleaseMessageBuffer](#) - Release an unused "zero copy" buffer pointer.
 - [CFE_SB_TransmitBuffer](#) - Transmit a buffer.
- [cFE Message Characteristics APIs](#)
 - [CFE_SB_SetUserDataLength](#) - Sets the length of user data in a software bus message.
 - [CFE_SB_TimeStampMsg](#) - Sets the time field in a software bus message with the current spacecraft time.
 - [CFE_SB_MessageStringSet](#) - Copies a string into a software bus message.
 - [CFE_SB_GetUserData](#) - Get a pointer to the user data portion of a software bus message.
 - [CFE_SB_GetUserDataLength](#) - Gets the length of user data in a software bus message.
 - [CFE_SB_MessageStringGet](#) - Copies a string out of a software bus message.
- [cFE Message ID APIs](#)
 - [CFE_SB_IsValidMsgId](#) - Identifies whether a given [CFE_SB_MsgId_t](#) is valid.
 - [CFE_SB_MsgId_Equal](#) - Identifies whether two [CFE_SB_MsgId_t](#) values are equal.
 - [CFE_SB_MsgIdToValue](#) - Converts a [CFE_SB_MsgId_t](#) to a normal integer.
 - [CFE_SB_ValueToMsgId](#) - Converts a normal integer into a [CFE_SB_MsgId_t](#).

4.7 Table Services API

- [cFE Registration APIs](#)
 - [CFE_TBL_Register](#) - Register a table with cFE to obtain Table Management Services.
 - [CFE_TBL_Share](#) - Obtain handle of table registered by another application.
 - [CFE_TBL_Unregister](#) - Unregister a table.
- [cFE Manage Table Content APIs](#)
 - [CFE_TBL_Load](#) - Load a specified table with data from specified source.
 - [CFE_TBL_Update](#) - Update contents of a specified table, if an update is pending.
 - [CFE_TBL_Validate](#) - Perform steps to validate the contents of a table image.
 - [CFE_TBL_Manage](#) - Perform standard operations to maintain a table.
 - [CFE_TBL_DumpToBuffer](#) - Copies the contents of a Dump Only Table to a shared buffer.
 - [CFE_TBL_Modified](#) - Notify cFE Table Services that table contents have been modified by the Application.
- [cFE Access Table Content APIs](#)

- [CFE_TBL_GetAddress](#) - Obtain the current address of the contents of the specified table.
- [CFE_TBL_GetAddresses](#) - Obtain the current addresses of an array of specified tables.
- [CFE_TBL_ReleaseAddress](#) - Release previously obtained pointer to the contents of the specified table.
- [CFE_TBL_ReleaseAddresses](#) - Release the addresses of an array of specified tables.
- cFE Get Table Information APIs
 - [CFE_TBL_GetStatus](#) - Obtain current status of pending actions for a table.
 - [CFE_TBL_GetInfo](#) - Obtain characteristics/information of/about a specified table.
 - [CFE_TBL_NotifyByMessage](#) - Instruct cFE Table Services to notify Application via message when table requires management.

4.8 Time Services API

- cFE Get Current Time APIs
 - [CFE_TIME_GetTime](#) - Get the current spacecraft time.
 - [CFE_TIME_GetTAI](#) - Get the current TAI (MET + SCTF) time.
 - [CFE_TIME_GetUTC](#) - Get the current UTC (MET + SCTF - Leap Seconds) time.
 - [CFE_TIME_GetMET](#) - Get the current value of the Mission Elapsed Time (MET).
 - [CFE_TIME_GetMETseconds](#) - Get the current seconds count of the mission-elapsed time.
 - [CFE_TIME_GetMETsubsecs](#) - Get the current sub-seconds count of the mission-elapsed time.
- cFE Get Time Information APIs
 - [CFE_TIME_GetSTCF](#) - Get the current value of the spacecraft time correction factor (STCF).
 - [CFE_TIME_GetLeapSeconds](#) - Get the current value of the leap seconds counter.
 - [CFE_TIME_GetClockState](#) - Get the current state of the spacecraft clock.
 - [CFE_TIME_GetClockInfo](#) - Provides information about the spacecraft clock.
- cFE Time Arithmetic APIs
 - [CFE_TIME_Add](#) - Adds two time values.
 - [CFE_TIME_Subtract](#) - Subtracts two time values.
 - [CFE_TIME_Compare](#) - Compares two time values.
- cFE Time Conversion APIs
 - [CFE_TIME_MET2SCTime](#) - Convert specified MET into Spacecraft Time.
 - [CFE_TIME_Sub2MicroSecs](#) - Converts a sub-seconds count to an equivalent number of microseconds.
 - [CFE_TIME_Micro2SubSecs](#) - Converts a number of microseconds to an equivalent sub-seconds count.
- cFE External Time Source APIs
 - [CFE_TIME_ExternalTone](#) - Provides the 1 Hz signal from an external source.
 - [CFE_TIME_ExternalMET](#) - Provides the Mission Elapsed Time from an external source.
 - [CFE_TIME_ExternalGPS](#) - Provide the time from an external source that has data common to GPS receivers.
 - [CFE_TIME_ExternalTime](#) - Provide the time from an external source that measures time relative to a known epoch.
 - [CFE_TIME_RegisterSyncCallback](#) - Registers a callback function that is called whenever time synchronization occurs.

- [CFE_TIME_UnregisterSynchCallback](#) - Unregisters a callback function that is called whenever time synchronization occurs.
- [cFE Miscellaneous Time APIs](#)
 - [CFE_TIME_Print](#) - Print a time value as a string.
 - [CFE_TIME_Local1HzISR](#) - This function is called via a timer callback set up at initialization of the TIME service.

5 Osal API Documentation

- General Information and Concepts
 - [OSAL Introduction](#)
- Core
 - [OSAL Return Code Defines](#)
 - [OSAL Object Type Defines](#)
 - APIs
 - * [OSAL Core Operation APIs](#)
 - * [OSAL Object ID Utility APIs](#)
 - * [OSAL Task APIs](#)
 - * [OSAL Message Queue APIs](#)
 - * [OSAL Heap APIs](#)
 - * [OSAL Error Info APIs](#)
 - * [OSAL Select APIs](#)
 - * [OSAL Printf APIs](#)
 - * [OSAL BSP low level access APIs](#)
 - * [OSAL Real Time Clock APIs](#)
 - * [OSAL Shell APIs](#)
 - [Common Reference](#)
 - [Return Code Reference](#)
 - [Id Map Reference](#)
 - [Clock Reference](#)
 - [Task Reference](#)
 - [Message Queue Reference](#)
 - [Heap Reference](#)
 - [Select Reference](#)
 - [Printf Reference](#)
 - [BSP Reference](#)
 - [Shell Reference](#)
- File System
 - [File System Overview](#)
 - [File Descriptors In Osal](#)
 - [OSAL File Access Option Defines](#)
 - [OSAL Reference Point For Seek Offset Defines](#)

- APIs
 - * OSAL Standard File APIs
 - * OSAL Directory APIs
 - * OSAL File System Level APIs
- File System Reference
- File Reference
- Directory Reference
- Object File Loader
 - APIs
 - * OSAL Dynamic Loader and Symbol APIs
 - File Loader Reference
- Network
 - APIs
 - * OSAL Network ID APIs
 - * OSAL Socket Address APIs
 - * OSAL Socket Management APIs
 - Network Reference
 - Socket Reference
- Timer
 - Timer Overview
 - APIs
 - * OSAL Time Base APIs
 - * OSAL Timer APIs
 - Timer Reference
 - Time Base Reference
- Semaphore and Mutex
 - OSAL Semaphore State Defines
 - APIs
 - * OSAL Binary Semaphore APIs
 - * OSAL Counting Semaphore APIs
 - * OSAL Mutex APIs
 - Binary Semaphore Reference
 - Counting Semaphore Reference
 - Mutex Reference

5.1 OSAL Introduction

The goal of this library is to promote the creation of portable and reusable real time embedded system software. Given the necessary OS abstraction layer implementations, the same embedded software should compile and run on a number of platforms ranging from spacecraft computer systems to desktop PCs.

The OS Application Program Interfaces (APIs) are broken up into core, file system, loader, network, and timer APIs. See the related document sections for full descriptions.

Note

The majority of these APIs should be called from a task running in the context of an OSAL application and in general should not be called from an ISR. There are a few exceptions, such as the ability to give a binary semaphore from an ISR.

5.2 File System Overview

The File System API is a thin wrapper around a selection of POSIX file APIs. In addition the File System API presents a common directory structure and volume view regardless of the underlying system type. For example, vxWorks uses MS-DOS style volume names and directories where a vxWorks RAM disk might have the volume "RAM:0". With this File System API, volumes are represented as Unix-style paths where each volume is mounted on the root file system:

- RAM:0/file1.dat becomes /mnt/ram/file1.dat
- FL:0/file2.dat becomes /mnt/fl/file2.dat

This abstraction allows the applications to use the same paths regardless of the implementation and it also allows file systems to be simulated on a desktop system for testing. On a desktop Linux system, the file system abstraction can be set up to map virtual devices to a regular directory. This is accomplished through the OS_mkfs call, OS_mount call, and a BSP specific volume table that maps the virtual devices to real devices or underlying file systems.

In order to make this file system volume abstraction work, a "Volume Table" needs to be provided in the Board Support Package of the application. The table has the following fields:

- Device Name: This is the name of the virtual device that the Application uses. Common names are "ramdisk1", "flash1", or "volatile1" etc. But the name can be any unique string.
- Physical Device Name: This is an implementation specific field. For vxWorks it is not needed and can be left blank. For a File system based implementation, it is the "mount point" on the root file system where all of the volume will be mounted. A common place for this on Linux could be a user's home directory, "/tmp", or even the current working directory "..". In the example of "/tmp" all of the directories created for the volumes would be under "/tmp" on the Linux file system. For a real disk device in Linux, such as a RAM disk, this field is the device name "/dev/ram0".
- Volume Type: This field defines the type of volume. The types are: FS_BASED which uses the existing file system, RAM_DISK which uses a RAM_DISK device in vxWorks, RTEMS, or Linux, FLASH_DISK_FORMAT which uses a flash disk that is to be formatted before use, FLASH_DISK_INIT which uses a flash disk with an existing format that is just to be initialized before its use, EEPROM which is for an EEPROM or PROM based system.
- Volatile Flag: This flag indicates that the volume or disk is a volatile disk (RAM disk) or a non-volatile disk, that retains its contents when the system is rebooted. This should be set to TRUE or FALSE.
- Free Flag: This is an internal flag that should be set to FALSE or zero.
- Is Mounted Flag: This is an internal flag that should be set to FALSE or zero. Note that a "pre-mounted" FS_BASED path can be set up by setting this flag to one.
- Volume Name: This is an internal field and should be set to a space character " ".
- Mount Point Field: This is an internal field and should be set to a space character " ".
- Block Size Field: This is used to record the block size of the device and does not need to be set by the user.

5.3 File Descriptors In Osal

The OSAL uses abstracted file descriptors. This means that the file descriptors passed back from the OS_open and OS_creat calls will only work with other OSAL OS_* calls. The reasoning for this is as follows:

Because the OSAL now keeps track of all file descriptors, OSAL specific information can be associated with a specific file descriptor in an OS independent way. For instance, the path of the file that the file descriptor points to can be easily retrieved. Also, the OSAL task ID of the task that opened the file can also be retrieved easily. Both of these pieces of information are very useful when trying to determine statistics for a task, or the entire system. This information can all be retrieved with a single API, OS_FDGetInfo.

All of the possible file system calls are not implemented. "Special" files requiring OS specific control/operations are by nature not portable. Abstraction in this case is not possible, so the raw OS calls should be used (including

open/close/etc). Mixing with OSAL calls is not supported for such cases. [OS_TranslatePath](#) is available to support using open directly by an app and maintain abstraction on the file system.

There are some small drawbacks with the OSAL file descriptors. Because the related information is kept in a table, there is a define called OS_MAX_NUM_OPEN_FILES that defines the maximum number of file descriptors available. This is a configuration parameter, and can be changed to fit your needs.

Also, if you open or create a file not using the OSAL calls (OS_open or OS_creat) then none of the other OS_* calls that accept a file descriptor as a parameter will work (the results of doing so are undefined). Therefore, if you open a file with the underlying OS's open call, you must continue to use the OS's calls until you close the file descriptor. Be aware that by doing this your software may no longer be OS agnostic.

5.4 Timer Overview

The timer API is a generic interface to the OS timer facilities. It is implemented using the POSIX timers on Linux and vxWorks and the native timer API on RTEMS. The number of timers supported is controlled by the configuration parameter OS_MAX_TIMERS.

6 cFE Mission Configuration Parameters

Global [CFE_MISSION_ES_CMD_TOPICID](#)

cFE Portable Message Numbers for Commands

Global [CFE_MISSION_ES_HK_TLM_TOPICID](#)

cFE Portable Message Numbers for Telemetry

Global [CFE_MISSION_EVS_CMD_TOPICID](#)

cFE Portable Message Numbers for Commands

Global [CFE_MISSION_EVS_HK_TLM_TOPICID](#)

cFE Portable Message Numbers for Telemetry

Global [CFE_MISSION_MAX_API_LEN](#)

cFE Maximum length for API names within data exchange structures

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

cFE Maximum length for filenames within data exchange structures

Global [CFE_MISSION_MAX_NUM_FILES](#)

cFE Maximum number of files in a message/data exchange

cFE Maximum number of files in a message/data exchange

Global [CFE_MISSION_MAX_PATH_LEN](#)

cFE Maximum length for pathnames within data exchange structures

cFE Maximum length for pathnames within data exchange structures

Global [CFE_MISSION_SB_CMD_TOPICID](#)

cFE Portable Message Numbers for Commands

Global [CFE_MISSION_SB_HK_TLM_TOPICID](#)

cFE Portable Message Numbers for Telemetry

Global [CFE_MISSION_TBL_CMD_TOPICID](#)

cFE Portable Message Numbers for Commands

- Global CFE_MISSION_TBL_HK_TLM_TOPICID**
cFE Portable Message Numbers for Telemetry
- Global CFE_MISSION_TIME_CMD_TOPICID**
cFE Portable Message Numbers for Commands
- Global CFE_MISSION_TIME_DATA_CMD_TOPICID**
cFE Portable Message Numbers for Global Messages
- Global CFE_MISSION_TIME_HK_TLM_TOPICID**
cFE Portable Message Numbers for Telemetry

7 Module Index

7.1 Modules

Here is a list of all modules:

CFS CFDP Event IDs	152
CFS CFDP Mission Configuration	192
CFS CFDP Version	194
CFS CFDP Command Codes	195
CFS CFDP Platform Configuration	216
CFS CFDP Telemetry	222
CFS CFDP Command Structures	224
CFS CFDP Command Message IDs	231
CFS CFDP Telemetry Message IDs	232
CFS CFDP Data Interface Message IDs	233
cFE Return Code Defines	234
cFE Resource ID APIs	257
cFE Entry/Exit APIs	260
cFE Application Control APIs	262
cFE Application Behavior APIs	265
cFE Information APIs	269
cFE Child Task APIs	278
cFE Miscellaneous APIs	282
cFE Critical Data Store APIs	285
cFE Memory Manager APIs	290

cFE Performance Monitor APIs	297
cFE Generic Counter APIs	299
cFE Registration APIs	305
cFE Send Event APIs	307
cFE Reset Event Filter APIs	312
cFE File Header Management APIs	314
cFE File Utility APIs	318
cFE Generic Message APIs	323
cFE Message Primary Header APIs	324
cFE Message Extended Header APIs	333
cFE Message Secondary Header APIs	339
cFE Message Id APIs	344
cFE Message Integrity APIs	346
cFE Pipe Management APIs	348
cFE Message Subscription Control APIs	353
cFE Send/Receive Message APIs	358
cFE Zero Copy APIs	361
cFE Message Characteristics APIs	364
cFE Message ID APIs	369
cFE SB Pipe options	374
cFE Registration APIs	375
cFE Manage Table Content APIs	380
cFE Access Table Content APIs	386
cFE Get Table Information APIs	391
cFE Table Type Defines	394
cFE Get Current Time APIs	396
cFE Get Time Information APIs	399
cFE Time Arithmetic APIs	402
cFE Time Conversion APIs	405
cFE External Time Source APIs	407

cFE Miscellaneous Time APIs	412
cFE Resource ID base values	415
cFE Clock State Flag Defines	417
OSAL Semaphore State Defines	419
OSAL Binary Semaphore APIs	420
OSAL BSP low level access APIs	425
OSAL Real Time Clock APIs	426
OSAL Core Operation APIs	440
OSAL Condition Variable APIs	444
OSAL Counting Semaphore APIs	450
OSAL Directory APIs	455
OSAL Return Code Defines	459
OSAL Error Info APIs	466
OSAL File Access Option Defines	468
OSAL Reference Point For Seek Offset Defines	469
OSAL Standard File APIs	470
OSAL File System Level APIs	483
OSAL Heap APIs	491
OSAL Object Type Defines	492
OSAL Object ID Utility APIs	495
OSAL Dynamic Loader and Symbol APIs	500
OSAL Mutex APIs	504
OSAL Network ID APIs	508
OSAL Printf APIs	510
OSAL Message Queue APIs	511
OSAL Select APIs	515
OSAL Shell APIs	521
OSAL Socket Address APIs	522
OSAL Socket Management APIs	526
OSAL Task APIs	536

OSAL Time Base APIs	542
OSAL Timer APIs	547

8 Data Structure Index

8.1 Data Structures

Here are the data structures with brief descriptions:

CCSDS_ExtendedHeader CCSDS packet extended header	553
CCSDS_PrimaryHeader CCSDS packet primary header	553
CF_AbandonCmd Abandon command structure	554
CF_AppData_t The CF application global state structure	555
CF_CancelCmd Cancel command structure	556
CF_CFDP_CycleTx_args Structure for use with the CF_CFDP_CycleTx() function	557
CF_CFDP_FileDirectiveDispatchTable_t A table of receive handler functions based on file directive code	557
CF_CFDP_Inv Structure representing CFDP LV Object format	558
CF_CFDP_PduAck Structure representing CFDP Acknowledge PDU	558
CF_CFDP_PduEof Structure representing CFDP End of file PDU	559
CF_CFDP_PduFileDataContent PDU file data content typedef for limit checking <code>outgoing_file_chunk_size</code> table value and set parameter command	560
CF_CFDP_PduFileDataHeader PDU file data header	560
CF_CFDP_PduFileDirectiveHeader Structure representing CFDP File Directive Header	561
CF_CFDP_PduFin Structure representing CFDP Finished PDU	561
CF_CFDP_PduHeader Structure representing base CFDP PDU header	562

CF_CFDP_PduMd	Structure representing CFDP Metadata PDU	563
CF_CFDP_PduNak	Structure representing CFDP Non-Acknowledge PDU	563
CF_CFDP_R_SubstateDispatchTable_t	A dispatch table for receive file transactions, receive side	564
CF_CFDP_S_SubstateRecvDispatchTable_t	A dispatch table for send file transactions, receive side	564
CF_CFDP_S_SubstateSendDispatchTable_t	A dispatch table for send file transactions, transmit side	565
CF_CFDP_SegmentRequest	Structure representing CFDP Segment Request	565
CF_CFDP_Tick_args	Structure for use with the CF_CFDP_DoTick() function	566
CF_CFDP_tlv	Structure representing CFDP TLV Object format	567
CF_CFDP_TxnRecvDispatchTable_t	A table of receive handler functions based on transaction state	568
CF_CFDP_TxnSendDispatchTable_t	A table of transmit handler functions based on transaction state	568
CF_CFDP_uint16_t	Encoded 16-bit value in the CFDP PDU	569
CF_CFDP_uint32_t	Encoded 32-bit value in the CFDP PDU	569
CF_CFDP_uint64_t	Encoded 64-bit value in the CFDP PDU	570
CF_CFDP_uint8_t	Encoded 8-bit value in the CFDP PDU	570
CF_ChAction_BoolArg	An object to use with channel-scope actions requiring only a boolean argument	570
CF_ChAction_BoolMsgArg	An object to use with channel-scope actions that require the message value	571
CF_ChAction_MsgArg	An object to use with channel-scope actions that require the message value	571
CF_ChAction_SuspResArg	An object to use with channel-scope actions for suspend/resume	572
CF_Channel	Channel state object	573

CF_ChannelConfig	Configuration entry for CFDP channel	574
CF_Chunk	Pairs an offset with a size to identify a specific piece of a file	577
CF_ChunkList	A list of CF_Chunk_t pairs	577
CF_ChunkWrapper	Wrapper around a CF_ChunkList_t object	578
CF_CListNode	Node link structure	579
CF_Codec_BitField		579
CF_CodecState	Tracks the current state of an encode or decode operation	580
CF_ConfigTable	Top-level CFDP configuration structure	581
CF_Crc	CRC state object	582
CF_DecoderState	Current state of a decode operation	583
CF_DisableDequeueCmd	DisableDequeue command structure	584
CF_DisableDirPollingCmd	DisableDirPolling command structure	584
CF_DisableEngineCmd	DisableEngine command structure	585
CF_EnableDequeueCmd	EnableDequeue command structure	585
CF_EnableDirPollingCmd	EnableDirPolling command structure	586
CF_EnableEngineCmd	EnableEngine command structure	587
CF_EncoderState	Current state of an encode operation	587
CF_Engine	An engine represents a pairing to a local EID	588
CF_EotPacket	End of transaction packet	590

CF_EotPacket_Payload	End of transaction packet	590
CF_Flags_Common	Data that applies to all types of transactions	592
CF_Flags_Rx	Flags that apply to receive transactions	593
CF_Flags_Tx	Flags that apply to send transactions	595
CF_FreezeCmd	Freeze command structure	595
CF_GapComputeArgs_t	Argument for Gap Compute function	596
CF_GetParam_Payload	Get parameter command structure	597
CF_GetParamCmd	Get parameter command structure	597
CF_History	CF History entry	598
CF_HkChannel_Data	Housekeeping channel data	600
CF_HkCmdCounters	Housekeeping command counters	601
CF_HkCounters	Housekeeping counters	602
CF_HkFault	Housekeeping fault counters	603
CF_HkPacket	Housekeeping packet	605
CF_HkPacket_Payload	Housekeeping packet	606
CF_HkRecv	Housekeeping received counters	607
CF_HkSent	Housekeeping sent counters	608
CF_Input	CF engine input state	609
CF_Logical_IntHeader	A union of all possible internal header types in a PDU	610

CF_Logical_Lv Structure representing logical LV Object format	611
CF_Logical_PduAck Structure representing CFDP Acknowledge PDU	612
CF_Logical_PduBuffer Encapsulates the entire PDU information	613
CF_Logical_PduEof Structure representing logical End of file PDU	615
CF_Logical_PduFileDataHeader	615
CF_Logical_PduFileDirectiveHeader Structure representing logical File Directive header	617
CF_Logical_PduFin Structure representing logical Finished PDU	617
CF_Logical_PduHeader Structure representing base CFDP PDU header	618
CF_Logical_PduMd Structure representing CFDP Metadata PDU	621
CF_Logical_PduNak Structure representing logical Non-Acknowledge PDU	622
CF_Logical_SegmentList	623
CF_Logical_SegmentRequest Structure representing logical Segment Request data	623
CF_Logical_Tlv Structure representing logical TLV Object format	624
CF_Logical_TlvData Union of various data items that may occur in a TLV item	625
CF_Logical_TlvList	626
CF_NoopCmd Noop command structure	626
CF_Output CF engine output state	627
CF_PduCmdMsg PDU command encapsulation structure	627
CF_PduTlmMsg PDU send encapsulation structure	628
CF_Playback CF Playback entry	629

CF_PlaybackDirCmd	Playback directory command structure	631
CF_Poll	CF Poll entry	631
CF_PollDir	Configuration entry for directory polling	632
CF_PurgeQueueCmd	PurgeQueue command structure	633
CF_ResetCountersCmd	Reset command structure	634
CF_ResumeCmd	Resume command structure	635
CF_RxS2_Data	Data specific to a class 2 receive file transaction	635
CF_RxState_Data	Data specific to a receive file transaction	636
CF_SendHkCmd	Send Housekeeping Command	637
CF_SetParam_Payload	Set parameter command structure	638
CF_SetParamCmd	Set parameter command structure	639
CF_StateData	Summary of all possible transaction state information (tx and rx)	639
CF_StateFlags	Summary of all possible transaction flags (tx and rx)	640
CF_SuspendCmd	Suspend command structure	641
CF_ThawCmd	Thaw command structure	642
CF_Timer	Basic CF timer object	642
CF_Transaction	Transaction state object	643
CF_Transaction_Payload	Transaction command structure	647
CF_Traverse_PriorityArg	Argument structure for use with CF_CList_Traverse_R()	648

CF_Traverse_TransSeqArg Argument structure for use with <code>CList_Traverse()</code>	648
CF_Traverse_WriteHistoryFileArg Argument structure for use with <code>CF_Traverse_WriteHistoryQueueEntryToFile()</code>	649
CF_Traverse_WriteTxnFileArg Argument structure for use with <code>CF_Traverse_WriteTxnQueueEntryToFile()</code>	650
CF_TraverseAll_Arg Argument structure for use with <code>CF_TraverseAllTransactions()</code>	651
CF_TxFile_Payload Transmit file command structure	651
CF_TxFileCmd Transmit file command structure	653
CF_TxnFilenames Cache of source and destination filename	654
CF_TxS2_Data Data specific to a class 2 send file transaction	654
CF_TxState_Data Data specific to a send file transaction	655
CF_UnionArgs_Payload Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32	656
CF_WakeupCmd Wake Up Command	656
CF_WriteQueue_Payload Write Queue command structure	657
CF_WriteQueueCmd Write Queue command structure	658
CFE_Config_ArrayValue Wrapper type for array configuration	659
CFE_Config_IdNameEntry	659
CFE_Config_ValueBuffer	660
CFE_Config_ValueEntry	660
CFE_ES_AppInfo Application Information	661
CFE_ES_AppNameCmd_Payload Generic application name command payload	665
CFE_ES_AppReloadCmd_Payload Reload Application Command Payload	665

CFE_ES_BlockStats	
Block statistics	666
CFE_ES_CDSRegDumpRec	
CDS Register Dump Record	667
CFE_ES_ClearERLogCmd	
	668
CFE_ES_ClearSysLogCmd	
	668
CFE_ES_DeleteCDSCmd	
Delete Critical Data Store Command	669
CFE_ES_DeleteCDSCmd_Payload	
Delete Critical Data Store Command Payload	669
CFE_ES_DumpCDSRegistryCmd	
Dump CDS Registry Command	670
CFE_ES_DumpCDSRegistryCmd_Payload	
Dump CDS Registry Command Payload	670
CFE_ES_FileNameCmd	
Generic file name command	671
CFE_ES_FileNameCmd_Payload	
Generic file name command payload	672
CFE_ES_HousekeepingTIm	
	672
CFE_ES_HousekeepingTIm_Payload	
	673
CFE_ES_MemPoolStats	
Memory Pool Statistics	681
CFE_ES_MemStatsTIm	
	682
CFE_ES_NoopCmd	
	683
CFE_ES_OneAppTIm	
	683
CFE_ES_OneAppTIm_Payload	
	684
CFE_ES_OverWriteSysLogCmd	
Overwrite/Discard System Log Configuration Command Payload	684
CFE_ES_OverWriteSysLogCmd_Payload	
Overwrite/Discard System Log Configuration Command Payload	685
CFE_ES_PoolAlign	
Pool Alignment	685
CFE_ES_PoolStatsTIm_Payload	
	686
CFE_ES_QueryAllCmd	
	687
CFE_ES_QueryAllTasksCmd	
	687

CFE_ES_QueryOneCmd	688
CFE_ES_ReloadAppCmd Reload Application Command	688
CFE_ES_ResetCountersCmd	689
CFE_ES_ResetPRCountCmd	689
CFE_ES_RestartAppCmd	690
CFE_ES_RestartCmd Restart cFE Command	690
CFE_ES_RestartCmd_Payload Restart cFE Command Payload	691
CFE_ES_SendHkCmd	692
CFE_ES_SendMemPoolStatsCmd Send Memory Pool Statistics Command	692
CFE_ES_SendMemPoolStatsCmd_Payload Send Memory Pool Statistics Command Payload	693
CFE_ES_SetMaxPRCountCmd Set Maximum Processor Reset Count Command	693
CFE_ES_SetMaxPRCountCmd_Payload Set Maximum Processor Reset Count Command Payload	694
CFE_ES_SetPerfFilterMaskCmd Set Performance Analyzer Filter Mask Command	694
CFE_ES_SetPerfFilterMaskCmd_Payload Set Performance Analyzer Filter Mask Command Payload	695
CFE_ES_SetPerfTriggerMaskCmd Set Performance Analyzer Trigger Mask Command	696
CFE_ES_SetPerfTrigMaskCmd_Payload Set Performance Analyzer Trigger Mask Command Payload	696
CFE_ES_StartApp Start Application Command	697
CFE_ES_StartAppCmd_Payload Start Application Command Payload	697
CFE_ES_StartPerfCmd_Payload Start Performance Analyzer Command Payload	699
CFE_ES_StartPerfDataCmd Start Performance Analyzer Command	699
CFE_ES_StopAppCmd	700

CFE_ES_StopPerfCmd_Payload Stop Performance Analyzer Command Payload	700
CFE_ES_StopPerfDataCmd Stop Performance Analyzer Command	701
CFE_ES_TaskInfo Task Information	701
CFE_ES_WriteERLogCmd	703
CFE_ES_WriteSysLogCmd	704
CFE_EVS_AddEventFilterCmd	704
CFE_EVS_AppDataCmd_Payload Write Event Services Application Information to File Command Payload	705
CFE_EVS_AppNameBitMaskCmd_Payload Generic App Name and Bitmask Command Payload	705
CFE_EVS_AppNameCmd_Payload Generic App Name Command Payload	706
CFE_EVS_AppNameEventIDCmd_Payload Generic App Name and Event ID Command Payload	706
CFE_EVS_AppNameEventIDMaskCmd_Payload Generic App Name, Event ID, Mask Command Payload	707
CFE_EVS_AppTlmData	708
CFE_EVS_BinFilter Event message filter definition structure	709
CFE_EVS_BitMaskCmd_Payload Generic Bitmask Command Payload	709
CFE_EVS_ClearLogCmd	710
CFE_EVS_DeleteEventFilterCmd	711
CFE_EVS_DisableAppEventsCmd	711
CFE_EVS_DisableAppEventTypeCmd	712
CFE_EVS_DisableEventTypeCmd	712
CFE_EVS_DisablePortsCmd	713
CFE_EVS_EnableAppEventsCmd	713
CFE_EVS_EnableAppEventTypeCmd	714
CFE_EVS_EnableEventTypeCmd	715
CFE_EVS_EnablePortsCmd	715

CFE_EVS_HousekeepingTlm	716
CFE_EVS_HousekeepingTlm_Payload	716
CFE_EVS_LogFileCmd_Payload Write Event Log to File Command Payload	720
CFE_EVS_LongEventTlm	720
CFE_EVS_LongEventTlm_Payload	721
CFE_EVS_NoopCmd	722
CFE_EVS_PacketID	722
CFE_EVS_ResetAllFiltersCmd	723
CFE_EVS_ResetAppCounterCmd	724
CFE_EVS_ResetCountersCmd	724
CFE_EVS_ResetFilterCmd	725
CFE_EVS_SendHkCmd	725
CFE_EVS_SetEventFormatCode_Payload Set Event Format Mode Command Payload	726
CFE_EVS_SetEventFormatModeCmd Set Event Format Mode Command	727
CFE_EVS_SetFilterCmd	727
CFE_EVS_SetLogMode_Payload Set Log Mode Command Payload	728
CFE_EVS_SetLogModeCmd Set Log Mode Command	728
CFE_EVS_ShortEventTlm	729
CFE_EVS_ShortEventTlm_Payload	730
CFE_EVS_WriteAppDataFileCmd Write Event Services Application Information to File Command	730
CFE_EVS_WriteLogFileCmd Write Event Log to File Command	731
CFE_FS_FileWriteMetaData External Metadata/State object associated with background file writes	731
CFE_FS_Header Standard cFE File header structure definition	733
CFE_SB_AllSubscriptionsTlm	734
CFE_SB_AllSubscriptionsTlm_Payload	735

CFE_SB_DisableRouteCmd	736
CFE_SB_DisableSubReportingCmd	736
CFE_SB_EnableRouteCmd	737
CFE_SB_EnableSubReportingCmd	737
CFE_SB_HousekeepingTIm	738
CFE_SB_HousekeepingTIm_Payload	738
CFE_SB_Msg	
Software Bus generic message	742
CFE_SB_MsgId_t	
CFE_SB_MsgId_t type definition	743
CFE_SB_MsgMapFileEntry	
SB Map File Entry	743
CFE_SB_NoopCmd	744
CFE_SB_PipeDepthStats	
SB Pipe Depth Statistics	744
CFE_SB_PipeInfoEntry	
SB Pipe Information File Entry	745
CFE_SB_Qos_t	
Quality Of Service Type Definition	747
CFE_SB_ResetCountersCmd	748
CFE_SB_RouteCmd_Payload	
Enable/Disable Route Command Payload	748
CFE_SB_RoutingFileEntry	
SB Routing File Entry	749
CFE_SB_SendHkCmd	750
CFE_SB_SendPrevSubsCmd	750
CFE_SB_SendSbStatsCmd	751
CFE_SB_SingleSubscriptionTIm	751
CFE_SB_SingleSubscriptionTIm_Payload	752
CFE_SB_StatsTIm	753
CFE_SB_StatsTIm_Payload	753
CFE_SB_SubEntries	
SB Previous Subscriptions Entry	757

CFE_SB_WriteFileInfoCmd_Payload Write File Info Command Payload	758
CFE_SB_WriteMapInfoCmd	758
CFE_SB_WritePipeInfoCmd	759
CFE_SB_WriteRoutingInfoCmd	759
CFE_TBL_AbortLoadCmd Abort Load Command	760
CFE_TBL_AbortLoadCmd_Payload Abort Load Command Payload	760
CFE_TBL_ActivateCmd Activate Table Command	761
CFE_TBL_ActivateCmd_Payload Activate Table Command Payload	762
CFE_TBL_DeleteCDSCmd_Payload Delete Critical Table CDS Command Payload	762
CFE_TBL_DeleteCDSCmd Delete Critical Table CDS Command	763
CFE_TBL_DumpCmd	763
CFE_TBL_DumpCmd_Payload Dump Table Command Payload	764
CFE_TBL_DumpRegistryCmd Dump Registry Command	765
CFE_TBL_DumpRegistryCmd_Payload Dump Registry Command Payload	765
CFE_TBL_File_Hdr The definition of the header fields that are included in CFE Table Data files	766
CFE_TBL_FileDef Table File summary object	767
CFE_TBL_HousekeepingTlm	768
CFE_TBL_HousekeepingTlm_Payload	769
CFE_TBL_Info Table Info	773
CFE_TBL_LoadCmd Load Table Command	775
CFE_TBL_LoadCmd_Payload Load Table Command Payload	775

CFE_TBL_NoopCmd	776
CFE_TBL_NotifyCmd	776
CFE_TBL_NotifyCmd_Payload Table Management Notification Command Payload	777
CFE_TBL_ResetCountersCmd	777
CFE_TBL_SendHkCmd	778
CFE_TBL_SendRegistryCmd Send Table Registry Command	778
CFE_TBL_SendRegistryCmd_Payload Send Table Registry Command Payload	779
CFE_TBL_TableRegistryTlm	779
CFE_TBL_TblRegPacket_Payload	780
CFE_TBL_ValidateCmd Validate Table Command	783
CFE_TBL_ValidateCmd_Payload Validate Table Command Payload	784
CFE_TIME_AddAdjustCmd	784
CFE_TIME_AddDelayCmd	785
CFE_TIME_AddOneHzAdjustmentCmd	786
CFE_TIME_DiagnosticTlm	786
CFE_TIME_DiagnosticTlm_Payload	787
CFE_TIME_FakeToneCmd	795
CFE_TIME_HousekeepingTlm	795
CFE_TIME_HousekeepingTlm_Payload	796
CFE_TIME_LeapsCmd_Payload Set leap seconds command payload	799
CFE_TIME_NoopCmd	799
CFE_TIME_OneHzAdjustmentCmd_Payload Generic seconds, subseconds command payload	800
CFE_TIME_OneHzCmd	800
CFE_TIME_ResetCountersCmd	801
CFE_TIME_SendDiagnosticCmd	801
CFE_TIME_SendHkCmd	802

CFE_TIME_SetLeapSecondsCmd		
Set leap seconds command		802
CFE_TIME_SetMETCmd		803
CFE_TIME_SetSignalCmd		
Set tone signal source command		803
CFE_TIME_SetSourceCmd		
Set time data source command		804
CFE_TIME_SetStateCmd		
Set clock state command		804
CFE_TIME_SetSTCFCmd		805
CFE_TIME_SetTimeCmd		806
CFE_TIME_SignalCmd_Payload		
Set tone signal source command payload		806
CFE_TIME_SourceCmd_Payload		
Set time data source command payload		807
CFE_TIME_StateCmd_Payload		
Set clock state command payload		807
CFE_TIME_SubAdjustCmd		808
CFE_TIME_SubDelayCmd		808
CFE_TIME_SubOneHzAdjustmentCmd		809
CFE_TIME_SysTime		
Data structure used to hold system time values		810
CFE_TIME_TimeCmd_Payload		
Generic seconds, microseconds command payload		810
CFE_TIME_ToneDataCmd		
Time at tone data command		811
CFE_TIME_ToneDataCmd_Payload		
Time at tone data command payload		812
CFE_TIME_ToneSignalCmd		812
OS_bin_sem_prop_t		
OSAL binary semaphore properties		813
OS_condvar_prop_t		
OSAL condition variable properties		814
OS_count_sem_prop_t		
OSAL counting semaphore properties		814
os_dirent_t		
Directory entry		815

OS_FdSet	An abstract structure capable of holding several OSAL IDs	815
OS_file_prop_t	OSAL file properties	816
os_fsinfo_t	OSAL file system info	816
os_fstat_t	File system status	817
OS_heap_prop_t	OSAL heap properties	818
OS_module_address_t	OSAL module address properties	819
OS_module_prop_t	OSAL module properties	820
OS_mut_sem_prop_t	OSAL mutex properties	821
OS_queue_prop_t	OSAL queue properties	821
OS_SockAddr_t	Encapsulates a generic network address	822
OS_SockAddrData_t	Storage buffer for generic network address	822
OS_socket_prop_t	Encapsulates socket properties	823
OS_static_symbol_record_t	Associates a single symbol name with a memory address	824
OS_statvfs_t		825
OS_task_prop_t	OSAL task properties	825
OS_time_t	OSAL time interval structure	826
OS_timebase_prop_t	Time base properties	827
OS_timer_prop_t	Timer properties	827

9 File Index

9.1 File List

Here is a list of all files with brief descriptions:

apps/cf/config/default_cf_extern_typedefs.h	828
apps/cf/config/default_cf_fcncodes.h	831
apps/cf/config/default_cf_interface_cfg.h	831
apps/cf/config/default_cf_internal_cfg.h	832
apps/cf/config/default_cf_mission_cfg.h	833
apps/cf/config/default_cf_msg.h	833
apps/cf/config/default_cf_msgdefs.h	834
apps/cf/config/default_cf_msgids.h	836
apps/cf/config/default_cf_msgstruct.h	836
apps/cf/config/default_cf_platform_cfg.h	839
apps/cf/config/default_cf_tbl.h	840
apps/cf/config/default_cf_tbldefs.h	840
apps/cf/config/default_cf_tblstruct.h	841
apps/cf/config/default_cf_topicids.h	841
apps/cf/fsw/inc/cf_events.h	843
apps/cf/fsw/inc/cf_perfids.h	848
apps/cf/fsw/src/cf_app.c	849
apps/cf/fsw/src/cf_app.h	855
apps/cf/fsw/src/cf_assert.h	862
apps/cf/fsw/src/cf_cfdp.c	863
apps/cf/fsw/src/cf_cfdp.h	907
apps/cf/fsw/src/cf_cfdp_dispatch.c	947
apps/cf/fsw/src/cf_cfdp_dispatch.h	951
apps/cf/fsw/src/cf_cfdp_pdu.h	955
apps/cf/fsw/src/cf_cfdp_r.c	961
apps/cf/fsw/src/cf_cfdp_r.h	985
apps/cf/fsw/src/cf_cfdp_s.c	1008

apps/cf/fsw/src/cf_cfdp_s.h	1029
apps/cf/fsw/src/cf_cfdp_sbintf.c	1049
apps/cf/fsw/src/cf_cfdp_sbintf.h	1053
apps/cf/fsw/src/cf_cfdp_types.h	1058
apps/cf/fsw/src/cf_chunk.c	1065
apps/cf/fsw/src/cf_chunk.h	1075
apps/cf/fsw/src/cf_clist.c	1086
apps/cf/fsw/src/cf_clist.h	1091
apps/cf/fsw/src/cf_cmd.c	1097
apps/cf/fsw/src/cf_cmd.h	1129
apps/cf/fsw/src/cf_codec.c	1162
apps/cf/fsw/src/cf_codec.h	1191
apps/cf/fsw/src/cf_crc.c	1220
apps/cf/fsw/src/cf_crc.h	1222
apps/cf/fsw/src/cf_dispatch.c	1224
apps/cf/fsw/src/cf_dispatch.h	1226
apps/cf/fsw/src/cf_eds_dispatch.c	1228
apps/cf/fsw/src/cf_logical_pdu.h	1231
apps/cf/fsw/src/cf_timer.c	1235
apps/cf/fsw/src/cf_timer.h	1237
apps/cf/fsw/src/cf_utils.c	1240
apps/cf/fsw/src/cf_utils.h	1258
apps/cf/fsw/src/cf_verify.h	1280
apps/cf/fsw/src/cf_version.h	1280
apps/cf/fsw/tables/cf_def_config.c	1281
build/osal_public_api/inc/osconfig.h	1281
cfe/cmake/sample_defs/example_mission_cfg.h	1287
cfe/cmake/sample_defs/example_platform_cfg.h	1298
cfe/cmake/sample_defs/sample_perfids.h	1342
cfe/modules/config/fsw/inc/cfe_config_external.h	1344

cfe/modules/config/fsw/inc/cfe_config_init.h	1345
cfe/modules/config/fsw/inc/cfe_config_lookup.h	1345
cfe/modules/config/fsw/inc/cfe_config_nametable.h	1346
cfe/modules/config/fsw/inc/cfe_config_set.h	1346
cfe/modules/config/fsw/inc/cfe_config_table.h	1347
cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h	1348
cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h	1350
cfe/modules/core_api/config/default_cfe_mission_cfg.h	1352
cfe/modules/core_api/config/default_cfe_msgids.h	1353
cfe/modules/core_api/fsw/inc/cfe.h	1353
cfe/modules/core_api/fsw/inc/cfe_config.h	1353
cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h	1357
cfe/modules/core_api/fsw/inc/cfe_endian.h	1358
cfe/modules/core_api/fsw/inc/cfe_error.h	1359
cfe/modules/core_api/fsw/inc/cfe_es.h	1367
cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h	1371
cfe/modules/core_api/fsw/inc/cfe_evs.h	1376
cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h	1377
cfe/modules/core_api/fsw/inc/cfe_fs.h	1380
cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h	1380
cfe/modules/core_api/fsw/inc/cfe_msg.h	1383
cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h	1385
cfe/modules/core_api/fsw/inc/cfe_resourceid.h	1389
cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h	1396
cfe/modules/core_api/fsw/inc/cfe_sb.h	1396
cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h	1399
cfe/modules/core_api/fsw/inc/cfe_tbl.h	1402
cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h	1403
cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h	1406
cfe/modules/core_api/fsw/inc/cfe_time.h	1407

cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h	1409
cfe/modules/core_api/fsw/inc/cfe_version.h	1411
cfe/modules/es/config/default_cfe_es_extern_typedefs.h	1413
cfe/modules/es/config/default_cfe_es_fcncodes.h	1421
cfe/modules/es/config/default_cfe_es_interface_cfg.h	1442
cfe/modules/es/config/default_cfe_es_internal_cfg.h	1445
cfe/modules/es/config/default_cfe_es_mission_cfg.h	1466
cfe/modules/es/config/default_cfe_es_msg.h	1466
cfe/modules/es/config/default_cfe_es_msgdefs.h	1466
cfe/modules/es/config/default_cfe_es_msgids.h	1470
cfe/modules/es/config/default_cfe_es_msgstruct.h	1471
cfe/modules/es/config/default_cfe_es_platform_cfg.h	1475
cfe/modules/es/config/default_cfe_es_topicids.h	1476
cfe/modules/es/fsw/inc/cfe_es_eventids.h	1477
cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h	1502
cfe/modules/evs/config/default_cfe_evs_fcncodes.h	1505
cfe/modules/evs/config/default_cfe_evs_interface_cfg.h	1523
cfe/modules/evs/config/default_cfe_evs_internal_cfg.h	1524
cfe/modules/evs/config/default_cfe_evs_mission_cfg.h	1528
cfe/modules/evs/config/default_cfe_evs_msg.h	1528
cfe/modules/evs/config/default_cfe_evs_msgdefs.h	1528
cfe/modules/evs/config/default_cfe_evs_msgids.h	1532
cfe/modules/evs/config/default_cfe_evs_msgstruct.h	1533
cfe/modules/evs/config/default_cfe_evs_platform_cfg.h	1536
cfe/modules/evs/config/default_cfe_evs_topicids.h	1536
cfe/modules/evs/fsw/inc/cfe_evs_eventids.h	1537
cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h	1549
cfe/modules/fs/config/default_cfe_fs_filedef.h	1549
cfe/modules/fs/config/default_cfe_fs_interface_cfg.h	1551
cfe/modules/fs/config/default_cfe_fs_mission_cfg.h	1552

cfe/modules/msg/fsw/inc/ ccsds_hdr.h	1552
cfe/modules/resourceid/fsw/inc/ cfe_core_resourceid_basevalues.h	1553
cfe/modules/resourceid/fsw/inc/ cfe_resourceid_basevalue.h	1553
cfe/modules/sb/config/ default_cfe_sb_extern_typedefs.h	1554
cfe/modules/sb/config/ default_cfe_sb_fcncodes.h	1556
cfe/modules/sb/config/ default_cfe_sb_interface_cfg.h	1566
cfe/modules/sb/config/ default_cfe_sb_internal_cfg.h	1567
cfe/modules/sb/config/ default_cfe_sb_mission_cfg.h	1575
cfe/modules/sb/config/ default_cfe_sb_msg.h	1575
cfe/modules/sb/config/ default_cfe_sb_msgdefs.h	1576
cfe/modules/sb/config/ default_cfe_sb_msgids.h	1578
cfe/modules/sb/config/ default_cfe_sb_msgstruct.h	1580
cfe/modules/sb/config/ default_cfe_sb_platform_cfg.h	1582
cfe/modules/sb/config/ default_cfe_sb_topicids.h	1582
cfe/modules/sb/fsw/inc/ cfe_sb_eventids.h	1583
cfe/modules/tbl/config/ default_cfe_tbl_extern_typedefs.h	1603
cfe/modules/tbl/config/ default_cfe_tbl_fcncodes.h	1604
cfe/modules/tbl/config/ default_cfe_tbl_interface_cfg.h	1613
cfe/modules/tbl/config/ default_cfe_tbl_internal_cfg.h	1614
cfe/modules/tbl/config/ default_cfe_tbl_mission_cfg.h	1620
cfe/modules/tbl/config/ default_cfe_tbl_msg.h	1620
cfe/modules/tbl/config/ default_cfe_tbl_msgdefs.h	1621
cfe/modules/tbl/config/ default_cfe_tbl_msgids.h	1623
cfe/modules/tbl/config/ default_cfe_tbl_msgstruct.h	1624
cfe/modules/tbl/config/ default_cfe_tbl_platform_cfg.h	1626
cfe/modules/tbl/config/ default_cfe_tbl_topicids.h	1626
cfe/modules/tbl/fsw/inc/ cfe_tbl_eventids.h	1627
cfe/modules/time/config/ default_cfe_time_extern_typedefs.h	1647
cfe/modules/time/config/ default_cfe_time_fcncodes.h	1652
cfe/modules/time/config/ default_cfe_time_interface_cfg.h	1668

cfe/modules/time/config/default_cfe_time_internal_cfg.h	1672
cfe/modules/time/config/default_cfe_time_mission_cfg.h	1677
cfe/modules/time/config/default_cfe_time_msg.h	1678
cfe/modules/time/config/default_cfe_time_msgdefs.h	1678
cfe/modules/time/config/default_cfe_time_msgids.h	1680
cfe/modules/time/config/default_cfe_time_msgstruct.h	1682
cfe/modules/time/config/default_cfe_time_platform_cfg.h	1685
cfe/modules/time/config/default_cfe_time_topicids.h	1685
cfe/modules/time/fsw/inc/cfe_time_eventids.h	1687
osal/src/os/inc/common_types.h	1697
osal/src/os/inc/osapi-binsem.h	1702
osal/src/os/inc/osapi-bsp.h	1703
osal/src/os/inc/osapi-clock.h	1703
osal/src/os/inc/osapi-common.h	1706
osal/src/os/inc/osapi-condvar.h	1708
osal/src/os/inc/osapi-constants.h	1709
osal/src/os/inc/osapi-countsem.h	1710
osal/src/os/inc/osapi-dir.h	1710
osal/src/os/inc/osapi-error.h	1711
osal/src/os/inc/osapi-file.h	1714
osal/src/os/inc/osapi-fs.h	1717
osal/src/os/inc/osapi-heap.h	1719
osal/src/os/inc/osapi-idmap.h	1719
osal/src/os/inc/osapi-macros.h	1721
osal/src/os/inc/osapi-module.h	1722
osal/src/os/inc/osapi-mutex.h	1724
osal/src/os/inc/osapi-network.h	1724
osal/src/os/inc/osapi-printf.h	1725
osal/src/os/inc/osapi-queue.h	1725
osal/src/os/inc/osapi-select.h	1726

osal/src/os/inc/osapi-shell.h	1727
osal/src/os/inc/osapi-sockets.h	1727
osal/src/os/inc/osapi-task.h	1730
osal/src/os/inc/osapi-timebase.h	1732
osal/src/os/inc/osapi-timer.h	1733
osal/src/os/inc/osapi-version.h	1734
osal/src/os/inc/osapi.h	1737
psp/fsw/inc/cfe_psp.h	1738
psp/fsw/inc/cfe_psp_cache_api.h	1739
psp/fsw/inc/cfe_psp_cds_api.h	1739
psp/fsw/inc/cfe_psp_eepromaccess_api.h	1741
psp/fsw/inc/cfe_psp_error.h CFE PSP Error header	1744
psp/fsw/inc/cfe_psp_exception_api.h	1747
psp/fsw/inc/cfe_psp_id_api.h	1749
psp/fsw/inc/cfe_psp_memaccess_api.h	1749
psp/fsw/inc/cfe_psp_memrange_api.h	1753
psp/fsw/inc/cfe_psp_port_api.h	1759
psp/fsw/inc/cfe_psp_ssr_api.h	1761
psp/fsw/inc/cfe_psp_timertick_api.h	1762
psp/fsw/inc/cfe_psp_version_api.h	1764
psp/fsw/inc/cfe_psp_watchdog_api.h	1765

10 Module Documentation

10.1 CFS CFDP Event IDs

Macros

- #define `CF_INIT_INF_EID` (20)
CF Initialization Event ID.
- #define `CF_INIT_TBL_CHECK_REL_ERR_EID` (21)
CF Check Table Release Address Failed Event ID.
- #define `CF_INIT_TBL_CHECK_MAN_ERR_EID` (22)
CF Check Table Manage Failed Event ID.
- #define `CF_INIT_TBL_CHECK_GA_ERR_EID` (23)

- #define **CF_INIT_TBL_REG_ERR_EID** (24)
CF Check Table Get Address Failed Event ID.
- #define **CF_INIT_TBL_LOAD_ERR_EID** (25)
CF Table Registration At Initialization Failed Event ID.
- #define **CF_INIT_TBL_MANAGE_ERR_EID** (26)
CF Table Load At Initialization Failed Event ID.
- #define **CF_INIT_TBL_GETADDR_ERR_EID** (27)
CF Table Manage At Initialization Failed Event ID.
- #define **CF_MID_ERR_EID** (28)
CF Table Get Address At Initialization Failed Event ID.
- #define **CF_INIT_MSG_RECV_ERR_EID** (29)
CF Message ID Invalid Event ID.
- #define **CF_INIT_SEM_ERR_EID** (30)
CF SB Receive Buffer Failed Event ID.
- #define **CF_CR_CHANNEL_PIPE_ERR_EID** (31)
CF Channel Semaphore Initialization Failed Event ID.
- #define **CF_INIT_SUB_ERR_EID** (32)
CF Channel Create Pipe Failed Event ID.
- #define **CF_INIT_TPS_ERR_EID** (33)
CF Channel Message Subscription Failed Event ID.
- #define **CF_INIT_CRC_ALIGN_ERR_EID** (34)
CF Ticks Per Second Config Table Validation Failed Event ID.
- #define **CF_INIT_OUTGOING_SIZE_ERR_EID** (35)
CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.
- #define **CF_CR_PIPE_ERR_EID** (36)
CF Outgoing Chunk Size Config Table Validation Failed Event ID.
- #define **CF_PDU_MD_RECVD_INF_EID** (40)
CF Create SB Command Pipe at Initialization Failed Event ID.
- #define **CF_PDU_SHORT_HEADER_ERR_EID** (41)
CF Metadata PDU Received Event ID.
- #define **CF_PDU_MD_SHORT_ERR_EID** (43)
CF PDU Header Too Short Event ID.
- #define **CF_PDU_INVALID_SRC_LEN_ERR_EID** (44)
CF Metadata PDU Too Short Event ID.
- #define **CF_PDU_INVALID_DST_LEN_ERR_EID** (45)
CF Metadata PDU Source Filename Length Invalid Event ID.
- #define **CF_PDU_FD_SHORT_ERR_EID** (46)
CF Metadata PDU Destination Filename Length Invalid Event ID.
- #define **CF_PDU_EOF_SHORT_ERR_EID** (47)
CF File Data PDU Too Short Event ID.
- #define **CF_PDU_ACK_SHORT_ERR_EID** (48)
CF End-Of-File PDU Too Short Event ID.
- #define **CF_PDU_FIN_SHORT_ERR_EID** (49)
CF Acknowledgment PDU Too Short Event ID.
- #define **CF_PDU_NAK_SHORT_ERR_EID** (50)
CF Finished PDU Too Short Event ID.
- #define **CF_PDU_NAK_SHORT_ERR_EID** (50)
CF Negative Acknowledgment PDU Too Short Event ID.

- #define CF_PDU_FD_UNSUPPORTED_ERR_EID (54)
CF File Data PDU Unsupported Option Event ID.
- #define CF_PDU_LARGE_FILE_ERR_EID (55)
CF PDU Header Large File Flag Set Event ID.
- #define CF_PDU_TRUNCATION_ERR_EID (56)
CF PDU Header Field Truncation.
- #define CF_RESET_FREED_XACT_DBG_EID (59)
Attempt to reset a transaction that has already been freed.
- #define CF_CFDP_RX_DROPPED_ERR_EID (60)
CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.
- #define CF_CFDP_INVALID_DST_ERR_EID (61)
CF PDU Received With Invalid Destination Entity ID Event ID.
- #define CF_CFDP_IDLE_MD_ERR_EID (62)
CF Invalid Metadata PDU Received On Idle Transaction Event ID.
- #define CF_CFDP_FD_UNHANDLED_ERR_EID (63)
CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.
- #define CF_CFDP_MAX_CMD_TX_ERR_EID (64)
CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.
- #define CF_CFDP_OPENDIR_ERR_EID (65)
CF Playback/Polling Directory Open Failed Event ID.
- #define CF_CFDP_DIR_SLOT_ERR_EID (66)
CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.
- #define CF_CFDP_NO_MSG_ERR_EID (67)
CF No Message Buffer Available Event ID.
- #define CF_CFDP_CLOSE_ERR_EID (68)
CF Close File Failed Event ID.
- #define CF_CFDP_R_REQUEST_MD_INF_EID (70)
CF Requesting RX Metadata Event ID.
- #define CF_CFDP_R_TEMP_FILE_INF_EID (71)
CF Creating Temp File For RX Transaction Without Metadata PDU.
- #define CF_CFDP_R_NAK_LIMIT_ERR_EID (72)
CF RX Transaction NAK Limit Reached Event ID.
- #define CF_CFDP_R_ACK_LIMIT_ERR_EID (73)
CF RX Transaction ACK Limit Reached Event ID.
- #define CF_CFDP_R_CRC_ERR_EID (74)
CF RX Transaction CRC Mismatch Event ID.
- #define CF_CFDP_R_SEEK_FD_ERR_EID (75)
CF RX File Data PDU Seek Failed Event ID.
- #define CF_CFDP_R_SEEK_CRC_ERR_EID (76)
CF RX Class 2 CRC Seek Failed Event ID.
- #define CF_CFDP_R_WRITE_ERR_EID (77)
CF RX File Data PDU Write Failed Event ID.
- #define CF_CFDP_R_SIZE_MISMATCH_ERR_EID (78)
CF RX End-Of-File PDU File Size Mismatch Event ID.
- #define CF_CFDP_R_PDU_EOF_ERR_EID (79)
CF Invalid End-Of-File PDU Event ID.
- #define CF_CFDP_R_CREAT_ERR_EID (80)

- `#define CF_CFDP_R_PDU_FINACK_ERR_EID` (81)
CF RX Transaction File Create Failed Event ID.
- `#define CF_CFDP_R_EOF_MD_SIZE_ERR_EID` (82)
CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.
- `#define CF_CFDP_R_RENAME_ERR_EID` (83)
CF RX Class 2 Metadata PDU Size Mismatch Event ID.
- `#define CF_CFDP_R_OPEN_ERR_EID` (84)
CF RX Class 2 Metadata PDU File Open Failed Event ID.
- `#define CF_CFDP_R_PDU_MD_ERR_EID` (85)
CF Invalid Out-of-order Metadata PDU Received Event ID.
- `#define CF_CFDP_R_READ_ERR_EID` (86)
CF Class 2 CRC Read From File Failed Event ID.
- `#define CF_CFDP_R_DC_INV_ERR_EID` (87)
CF RX Invalid File Directive PDU Code Received Event ID.
- `#define CF_CFDP_R_INACT_TIMER_ERR_EID` (88)
CF RX Inactivity Timer Expired Event ID.
- `#define CF_CFDP_S_START_SEND_INF_EID` (90)
CF TX Initiated Event ID.
- `#define CF_CFDP_S_SEEK_FD_ERR_EID` (91)
CF TX File Data PDU Seek Failed Event ID.
- `#define CF_CFDP_S_READ_ERR_EID` (92)
CF TX File Data PDU Read Failed Event ID.
- `#define CF_CFDP_S_SEND_FD_ERR_EID` (93)
CF TX File Data PDU Send Failed Event ID.
- `#define CF_CFDP_S_ALREADY_OPEN_ERR_EID` (94)
CF TX Metadata PDU File Already Open Event ID.
- `#define CF_CFDP_S_OPEN_ERR_EID` (95)
CF TX Metadata PDU File Open Failed Event ID.
- `#define CF_CFDP_S_SEEK_END_ERR_EID` (96)
CF TX Metadata PDU File Seek End Failed Event ID.
- `#define CF_CFDP_S_SEEK_BEG_ERR_EID` (97)
CF TX Metadata PDU File Seek Beginning Failed Event ID.
- `#define CF_CFDP_S_SEND_MD_ERR_EID` (98)
CF TX Metadata PDU Send Failed Event ID.
- `#define CF_CFDP_S_INVALID_SR_ERR_EID` (100)
CF TX Received NAK PDU Bad Segment Request Event ID.
- `#define CF_CFDP_S_PDU_NAK_ERR_EID` (101)
CF TX Received NAK PDU Invalid Event ID.
- `#define CF_CFDP_S_PDU_EOF_ERR_EID` (102)
CF TX Received EOF ACK PDU Invalid Event ID.
- `#define CF_CFDP_S_EARLY_FIN_ERR_EID` (103)
CF TX Received Early FIN PDU Event ID.
- `#define CF_CFDP_S_DC_INV_ERR_EID` (104)
CF Invalid TX File Directive PDU Code Event ID.
- `#define CF_CFDP_S_NON_FD_PDU_ERR_EID` (105)
CF Received TX Non-File Directive PDU Event ID.

- #define CF_CFDP_S_ACK_LIMIT_ERR_EID (106)
 CF TX EOF PDU Send Limit Reached Event ID.
- #define CF_CFDP_S_INACT_TIMER_ERR_EID (107)
 CF TX Inactivity Timer Expired Event ID.
- #define CF_NOOP_INF_EID (110)
 CF NOOP Command Received Event ID.
- #define CF_RESET_INF_EID (111)
 CF Reset Counters Command Received Event ID.
- #define CF_CMD_GETSET1_INF_EID (112)
 CF Set Parameter Command Received Event ID.
- #define CF_CMD_GETSET2_INF_EID (113)
 CF Get Parameter Command Received Event ID.
- #define CF_CMD_SUSPRES_INF_EID (114)
 CF Suspend/Resume Command Received Event ID.
- #define CF_CMD_WQ_INF_EID (115)
 CF Write Queue Command Received Event ID.
- #define CF_CMD_ENABLE_ENGINE_INF_EID (116)
 CF Enable Engine Command Received Event ID.
- #define CF_CMD_DISABLE_ENGINE_INF_EID (117)
 CF Disable Engine Command Received Event ID.
- #define CF_CMD_TX_FILE_INF_EID (118)
 CF Transfer File Command Received Event ID.
- #define CF_CMD_PLAYBACK_DIR_INF_EID (119)
 CF Playback Directory Command Received Event ID.
- #define CF_CMD_FREEZE_INF_EID (120)
 CF Freeze Command Received Event ID.
- #define CF_CMD_THAW_INF_EID (121)
 CF Thaw Command Received Event ID.
- #define CF_CMD_CANCEL_INF_EID (122)
 CF Cancel Command Received Event ID.
- #define CF_CMD_ABANDON_INF_EID (123)
 CF Abandon Command Received Event ID.
- #define CF_CMD_ENABLE_DEQUEUE_INF_EID (124)
 CF Enable Dequeue Command Received Event ID.
- #define CF_CMD_DISABLE_DEQUEUE_INF_EID (125)
 CF Disable Dequeue Command Received Event ID.
- #define CF_CMD_ENABLE_POLLDIR_INF_EID (126)
 CF Enable Polldir Command Received Event ID.
- #define CF_CMD_DISABLE_POLLDIR_INF_EID (127)
 CF Disable Polldir Command Received Event ID.
- #define CF_CMD_PURGE_QUEUE_INF_EID (128)
 CF Purge Queue Command Received Event ID.
- #define CF_CMD_RESET_INVALID_ERR_EID (129)
 CF Reset Counters Command Invalid Event ID.
- #define CF_CMD_CHAN_PARAM_ERR_EID (130)
 CF Command Channel Invalid Event ID.
- #define CF_CMD_TRANS_NOT_FOUND_ERR_EID (131)

- `#define CF_CMD_TSN_CHAN_INVALID_ERR_EID` (132)
CF Command Transaction Invalid Event ID.
- `#define CF_CMD_SUSPRES_SAME_INF_EID` (133)
CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.
- `#define CF_CMD_SUSPRES_CHAN_ERR_EID` (134)
CF Suspend/Resume Command No Matching Transaction Event ID.
- `#define CF_CMD_POLLDIR_INVALID_ERR_EID` (135)
CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.
- `#define CF_CMD_PURGE_ARG_ERR_EID` (136)
CF Purge Queue Command Invalid Argument Event ID.
- `#define CF_CMD_WQ_CHAN_ERR_EID` (137)
CF Write Queue Command Invalid Channel Event ID.
- `#define CF_CMD_WQ_ARGS_ERR_EID` (138)
CF Write Queue Command Invalid Queue Event ID.
- `#define CF_CMD_WQ_OPEN_ERR_EID` (139)
CF Write Queue Command File Open Failed Event ID.
- `#define CF_CMD_WQ_WRITEQ_RX_ERR_EID` (140)
CF Write Queue Command RX Active File Write Failed Event ID.
- `#define CF_CMD_WQ_WRITEHIST_RX_ERR_EID` (141)
CF Write Queue Command RX History File Write Failed Event ID.
- `#define CF_CMD_WQ_WRITEQ_TX_ERR_EID` (142)
CF Write Queue Command TX Active File Write Failed Event ID.
- `#define CF_CMD_WQ_WRITEQ_PEND_ERR_EID` (143)
CF Write Queue Command TX Pending File Write Failed Event ID.
- `#define CF_CMD_WQ_WRITEHIST_TX_ERR_EID` (144)
CF Write Queue Command TX History File Write Failed Event ID.
- `#define CF_CMD_GETSET_VALIDATE_ERR_EID` (145)
CF Set Parameter Command Parameter Validation Failed Event ID.
- `#define CF_CMD_GETSET_PARAM_ERR_EID` (146)
CF Set/Get Parameter Command Invalid Parameter ID Event ID.
- `#define CF_CMD_GETSET_CHAN_ERR_EID` (147)
CF Set/Get Parameter Command Invalid Channel Event ID.
- `#define CF_CMD_ENABLE_ENGINE_ERR_EID` (148)
CF Enable Engine Command Failed Event ID.
- `#define CF_CMD_ENG_ALREADY_ENA_INF_EID` (149)
CF Enable Engine Command Engine Already Enabled Event ID.
- `#define CF_CMD_ENG_ALREADY_DIS_INF_EID` (150)
CF Disable Engine Command Engine Already Disabled Event ID.
- `#define CF_CMD_LEN_ERR_EID` (151)
CF Command Length Verification Failed Event ID.
- `#define CF_CC_ERR_EID` (152)
CF Command Code Invalid Event ID.
- `#define CF_CMD_WHIST_WRITE_ERR_EID` (153)
CF Write Entry To File Failed Event ID.
- `#define CF_CMD_BAD_PARAM_ERR_EID` (154)
CF Playback Dir Or TX File Command Bad Parameter Event ID.

- #define CF_CMD_CANCEL_CHAN_ERR_EID (155)
CF Cancel Command No Matching Transaction Event ID.
- #define CF_CMD_ABANDON_CHAN_ERR_EID (156)
CF Abandon Command No Matching Transaction Event ID.
- #define CF_CMD_TX_FILE_ERR_EID (157)
CF Transfer File Command Failed Event ID.
- #define CF_CMD_PLAYBACK_DIR_ERR_EID (158)
CF Playback Directory Command Failed Event ID.
- #define CF_CMD_FREEZE_ERR_EID (159)
CF Freeze Command Failed Event ID.
- #define CF_CMD_THAW_ERR_EID (160)
CF Thaw Command Failed Event ID.
- #define CF_CMD_ENABLE_DEQUEUE_ERR_EID (161)
CF Enable Dequeue Command Failed Event ID.
- #define CF_CMD_DISABLE_DEQUEUE_ERR_EID (162)
CF Disable Dequeue Command Failed Event ID.
- #define CF_CMD_ENABLE_POLLDIR_ERR_EID (163)
CF Enable Polldir Command Failed Event ID.
- #define CF_CMD_DISABLE_POLLDIR_ERR_EID (164)
CF Disable Polldir Command Failed Event ID.
- #define CF_CMD_PURGE_QUEUE_ERR_EID (165)
CF Purge Queue Command Failed Event ID.

10.1.1 Detailed Description

10.1.2 Macro Definition Documentation

10.1.2.1 CF_CC_ERR_EID #define CF_CC_ERR_EID (152)
CF Command Code Invalid Event ID.

Type: ERROR

Cause:

Received command code unrecognized
Definition at line 1386 of file cf_events.h.

10.1.2.2 CF_CFDP_CLOSE_ERR_EID #define CF_CFDP_CLOSE_ERR_EID (68)
CF Close File Failed Event ID.

Type: ERROR

Cause:

Failure from file close call
Definition at line 491 of file cf_events.h.

10.1.2.3 CF_CFDP_DIR_SLOT_ERR_EID #define CF_CFDP_DIR_SLOT_ERR_EID (66)
CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.

Type: ERROR

Cause:

Command request to playback a directory received when channel is already handling the maximum number of concurrent playback directories

Definition at line 469 of file cf_events.h.

10.1.2.4 CF_CFDP_FD_UNHANDLED_ERR_EID #define CF_CFDP_FD_UNHANDLED_ERR_EID (63)
CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.

Type: ERROR

Cause:

File Directive PDU received without the metadata directive code on an idle transaction

Definition at line 434 of file cf_events.h.

10.1.2.5 CF_CFDP_IDLE_MD_ERR_EID #define CF_CFDP_IDLE_MD_ERR_EID (62)
CF Invalid Metadata PDU Received On Idle Transaction Event ID.

Type: ERROR

Cause:

Metadata PDU received for an idle transaction failed decoding

Definition at line 423 of file cf_events.h.

10.1.2.6 CF_CFDP_INVALID_DST_ERR_EID #define CF_CFDP_INVALID_DST_ERR_EID (61)
CF PDU Received With Invalid Destination Entity ID Event ID.

Type: ERROR

Cause:

PDU without a matching/existing transaction received with an entity ID that doesn't match the receiving channel's entity ID

Definition at line 412 of file cf_events.h.

10.1.2.7 CF_CFDP_MAX_CMD_TX_ERR_EID #define CF_CFDP_MAX_CMD_TX_ERR_EID (64)
CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.

Type: ERROR

Cause:

Command request to transmit a file received when channel is already handling the maximum number of concurrent command transmit transactions

Definition at line 446 of file cf_events.h.

10.1.2.8 CF_CFDP_NO_MSG_ERR_EID #define CF_CFDP_NO_MSG_ERR_EID (67)
CF No Message Buffer Available Event ID.

Type: ERROR

Cause:

Failure from SB allocate message buffer call when constructing PDU

Definition at line 480 of file cf_events.h.

10.1.2.9 CF_CFDP_OPENDIR_ERR_EID #define CF_CFDP_OPENDIR_ERR_EID (65)
CF Playback/Polling Directory Open Failed Event ID.

Type: ERROR

Cause:

Failure opening directory during playback or polling initialization

Definition at line 457 of file cf_events.h.

10.1.2.10 CF_CFDP_R_ACK_LIMIT_ERR_EID #define CF_CFDP_R_ACK_LIMIT_ERR_EID (73)
CF RX Transaction ACK Limit Reached Event ID.

Type: ERROR

Cause:

Condition that triggers an ACK occurred that would meet or exceed the ACK limit

Definition at line 541 of file cf_events.h.

10.1.2.11 CF_CFDP_R_CRC_ERR_EID #define CF_CFDP_R_CRC_ERR_EID (74)
CF RX Transaction CRC Mismatch Event ID.

Type: ERROR

Cause:

RX Transaction final CRC mismatch
Definition at line 552 of file cf_events.h.

10.1.2.12 CF_CFDP_R_CREAT_ERR_EID #define CF_CFDP_R_CREAT_ERR_EID (80)
CF RX Transaction File Create Failed Event ID.

Type: ERROR

Cause:

Failure in opencreate file call for an RX transaction
Definition at line 619 of file cf_events.h.

10.1.2.13 CF_CFDP_R_DC_INV_ERR_EID #define CF_CFDP_R_DC_INV_ERR_EID (87)
CF RX Invalid File Directive PDU Code Received Event ID.

Type: ERROR

Cause:

Unrecognized file directive PDU directive code received for a current transaction
Definition at line 700 of file cf_events.h.

10.1.2.14 CF_CFDP_R_EOF_MD_SIZE_ERR_EID #define CF_CFDP_R_EOF_MD_SIZE_ERR_EID (82)
CF RX Class 2 Metadata PDU Size Mismatch Event ID.

Type: ERROR

Cause:

Out-of-order RX Class 2 Metadata PDU received with file size that doesn't match already received EOF PDU file size
Definition at line 642 of file cf_events.h.

10.1.2.15 CF_CFDP_R_INACT_TIMER_ERR_EID #define CF_CFDP_R_INACT_TIMER_ERR_EID (88)
CF RX Inactivity Timer Expired Event ID.

Type: ERROR

Cause:

Expiration of the RX inactivity timer
Definition at line 711 of file cf_events.h.

10.1.2.16 CF_CFDP_R_NAK_LIMIT_ERR_EID #define CF_CFDP_R_NAK_LIMIT_ERR_EID (72)
CF RX Transaction NAK Limit Reached Event ID.

Type: ERROR

Cause:

Condition that triggers a NAK occurred that would meet or exceed the NAK limit
Definition at line 530 of file cf_events.h.

10.1.2.17 CF_CFDP_R_OPEN_ERR_EID #define CF_CFDP_R_OPEN_ERR_EID (84)
CF RX Class 2 Metadata PDU File Open Failed Event ID.

Type: ERROR

Cause:

Failure from file open call after reception of an out-of-order RX Class 2 Metadata PDU
Definition at line 666 of file cf_events.h.

10.1.2.18 CF_CFDP_R_PDU_EOF_ERR_EID #define CF_CFDP_R_PDU_EOF_ERR_EID (79)
CF Invalid End-Of-File PDU Event ID.

Type: ERROR

Cause:

End-of-file PDU failed decoding
Definition at line 608 of file cf_events.h.

10.1.2.19 CF_CFDP_R_PDU_FINACK_ERR_EID #define CF_CFDP_R_PDU_FINACK_ERR_EID (81)
CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.

Type: ERROR

Cause:

ACK PDU failed decoding during Class 2 RX Transaction
Definition at line 630 of file cf_events.h.

10.1.2.20 CF_CFDP_R_PDU_MD_ERR_EID #define CF_CFDP_R_PDU_MD_ERR_EID (85)
CF Invalid Out-of-order Metadata PDU Received Event ID.

Type: ERROR

Cause:

Failure to decode out-of-order metadata PDU
Definition at line 677 of file cf_events.h.

10.1.2.21 CF_CFDP_R_READ_ERR_EID #define CF_CFDP_R_READ_ERR_EID (86)
CF Class 2 CRC Read From File Failed Event ID.

Type: ERROR

Cause:

Failure from file read call during RX Class 2 CRC calculation
Definition at line 688 of file cf_events.h.

10.1.2.22 CF_CFDP_R_RENAME_ERR_EID #define CF_CFDP_R_RENAME_ERR_EID (83)
CF RX Class 2 Metadata PDU File Rename Failed Event ID.

Type: ERROR

Cause:

Failure from file rename call after reception of an out-of-order RX Class 2 Metadata PDU
Definition at line 654 of file cf_events.h.

10.1.2.23 CF_CFDP_R_REQUEST_MD_INF_EID #define CF_CFDP_R_REQUEST_MD_INF_EID (70)
CF Requesting RX Metadata Event ID.

Type: INFORMATION

Cause:

RX transaction missing metadata which results in a NAK being sent to request a metadata PDU for the transaction
Definition at line 507 of file cf_events.h.

10.1.2.24 CF_CFDP_R_SEEK_CRC_ERR_EID #define CF_CFDP_R_SEEK_CRC_ERR_EID (76)
CF RX Class 2 CRC Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when calculating CRC from the file at the end of a Class 2 RX transaction
Definition at line 575 of file cf_events.h.

10.1.2.25 CF_CFDP_R_SEEK_FD_ERR_EID #define CF_CFDP_R_SEEK_FD_ERR_EID (75)
CF RX File Data PDU Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when processing out of order file data PDUs
Definition at line 563 of file cf_events.h.

10.1.2.26 CF_CFDP_R_SIZE_MISMATCH_ERR_EID #define CF_CFDP_R_SIZE_MISMATCH_ERR_EID (78)
CF RX End-Of-File PDU File Size Mismatch Event ID.

Type: ERROR

Cause:

End-of-file PDU file size does not match transaction expected file size
Definition at line 597 of file cf_events.h.

10.1.2.27 CF_CFDP_R_TEMP_FILE_INF_EID #define CF_CFDP_R_TEMP_FILE_INF_EID (71)
CF Creating Temp File For RX Transaction Without Metadata PDU.

Type: INFORMATION

Cause:

RX transaction missing metadata causing creation of a temporary filename to store the data
Definition at line 519 of file cf_events.h.

10.1.2.28 CF_CFDP_R_WRITE_ERR_EID #define CF_CFDP_R_WRITE_ERR_EID (77)
CF RX File Data PDU Write Failed Event ID.

Type: ERROR

Cause:

Failure of write to file call when processing file data PDUs
Definition at line 586 of file cf_events.h.

10.1.2.29 CF_CFDP_RX_DROPPED_ERR_EID #define CF_CFDP_RX_DROPPED_ERR_EID (60)
CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.

Type: ERROR

Cause:

PDU without a matching/existing transaction received when channel receive queue is already handling the maximum number of concurrent receive transactions
Definition at line 400 of file cf_events.h.

10.1.2.30 CF_CFDP_S_ACK_LIMIT_ERR_EID #define CF_CFDP_S_ACK_LIMIT_ERR_EID (106)
CF TX EOF PDU Send Limit Reached Event ID.

Type: ERROR

Cause:

Timed out the limit number of times waiting for an ACK PDU for the EOF PDU on a current transaction
Definition at line 898 of file cf_events.h.

10.1.2.31 CF_CFDP_S_ALREADY_OPEN_ERR_EID #define CF_CFDP_S_ALREADY_OPEN_ERR_EID (94)
CF TX Metadata PDU File Already Open Event ID.

Type: ERROR

Cause:

Failure to send metadata PDU due to file already being open
Definition at line 770 of file cf_events.h.

10.1.2.32 CF_CFDP_S_DC_INV_ERR_EID #define CF_CFDP_S_DC_INV_ERR_EID (104)
CF Invalid TX File Directive PDU Code Event ID.

Type: ERROR

Cause:

Unrecognized file directive PDU directive code received for a current transaction
Definition at line 875 of file cf_events.h.

10.1.2.33 CF_CFDP_S_EARLY_FIN_ERR_EID #define CF_CFDP_S_EARLY_FIN_ERR_EID (103)
CF TX Received Early FIN PDU Event ID.

Type: ERROR

Cause:

Early FIN PDU received prior to completion of a current transaction
Definition at line 863 of file cf_events.h.

10.1.2.34 CF_CFDP_S_INACT_TIMER_ERR_EID #define CF_CFDP_S_INACT_TIMER_ERR_EID (107)
CF TX Inactivity Timer Expired Event ID.

Type: ERROR

Cause:

Send transaction activity timeout expired
Definition at line 909 of file cf_events.h.

10.1.2.35 CF_CFDP_S_INVALID_SR_ERR_EID #define CF_CFDP_S_INVALID_SR_ERR_EID (100)
CF TX Received NAK PDU Bad Segment Request Event ID.

Type: ERROR

Cause:

Bad segment request values in received NAK PDU relating to a current transaction
Definition at line 828 of file cf_events.h.

10.1.2.36 CF_CFDP_S_NON_FD_PDU_ERR_EID #define CF_CFDP_S_NON_FD_PDU_ERR_EID (105)
CF Received TX Non-File Directive PDU Event ID.

Type: ERROR

Cause:

Received a non-file directive PDU on a send transaction
Definition at line 886 of file cf_events.h.

10.1.2.37 CF_CFDP_S_OPEN_ERR_EID #define CF_CFDP_S_OPEN_ERR_EID (95)
CF TX Metadata PDU File Open Failed Event ID.

Type: ERROR

Cause:

Failure in file open call when preparing to send metadata PDU
Definition at line 781 of file cf_events.h.

10.1.2.38 CF_CFDP_S_PDU_EOF_ERR_EID #define CF_CFDP_S_PDU_EOF_ERR_EID (102)
CF TX Received EOF ACK PDU Invalid Event ID.

Type: ERROR

Cause:

Failure processing received ACK PDU relating to a current transaction
Definition at line 852 of file cf_events.h.

10.1.2.39 CF_CFDP_S_PDU_NAK_ERR_EID #define CF_CFDP_S_PDU_NAK_ERR_EID (101)
CF TX Received NAK PDU Invalid Event ID.

Type: ERROR

Cause:

Failure processing received NAK PDU relating to a current transaction
Definition at line 840 of file cf_events.h.

10.1.2.40 CF_CFDP_S_READ_ERR_EID #define CF_CFDP_S_READ_ERR_EID (92)
CF TX File Data PDU Read Failed Event ID.

Type: ERROR

Cause:

Failure of read file call when preparing to send file data PDU
Definition at line 748 of file cf_events.h.

10.1.2.41 CF_CFDP_S_SEEK_BEG_ERR_EID #define CF_CFDP_S_SEEK_BEG_ERR_EID (97)
CF TX Metadata PDU File Seek Beginning Failed Event ID.

Type: ERROR

Cause:

Failure in file lseek to beginning of file call when preparing to send metadata PDU
Definition at line 805 of file cf_events.h.

10.1.2.42 CF_CFDP_S_SEEK_END_ERR_EID #define CF_CFDP_S_SEEK_END_ERR_EID (96)
CF TX Metadata PDU File Seek End Failed Event ID.

Type: ERROR

Cause:

Failure in file lseek to end of file call when preparing to send metadata PDU
Definition at line 793 of file cf_events.h.

10.1.2.43 CF_CFDP_S_SEEK_FD_ERR_EID #define CF_CFDP_S_SEEK_FD_ERR_EID (91)
CF TX File Data PDU Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when preparing to send file data PDU
Definition at line 737 of file cf_events.h.

10.1.2.44 CF_CFDP_S_SEND_FD_ERR_EID #define CF_CFDP_S_SEND_FD_ERR_EID (93)
CF TX File Data PDU Send Failed Event ID.

Type: ERROR

Cause:

Failure to send the file data PDU
Definition at line 759 of file cf_events.h.

10.1.2.45 CF_CFDP_S_SEND_MD_ERR_EID #define CF_CFDP_S_SEND_MD_ERR_EID (98)
CF TX Metadata PDU Send Failed Event ID.

Type: ERROR

Cause:

Failure to send the metadata PDU
Definition at line 816 of file cf_events.h.

10.1.2.46 CF_CFDP_S_START_SEND_INF_EID #define CF_CFDP_S_START_SEND_INF_EID (90)
CF TX Initiated Event ID.

Type: INFORMATION

Cause:

File TX transaction initiated
Definition at line 726 of file cf_events.h.

10.1.2.47 CF_CMD_ABANDON_CHAN_ERR_EID #define CF_CMD_ABANDON_CHAN_ERR_EID (156)
CF Abandon Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Abandon command received without a matching transaction
Definition at line 1430 of file cf_events.h.

10.1.2.48 CF_CMD_ABANDON_INF_EID #define CF_CMD_ABANDON_INF_EID (123)
CF Abandon Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of abandon command
Definition at line 1067 of file cf_events.h.

10.1.2.49 CF_CMD_BAD_PARAM_ERR_EID #define CF_CMD_BAD_PARAM_ERR_EID (154)
CF Playback Dir Or TX File Command Bad Parameter Event ID.

Type: ERROR

Cause:

Bad parameter received in playback directory or transfer file command
Definition at line 1408 of file cf_events.h.

10.1.2.50 CF_CMD_CANCEL_CHAN_ERR_EID #define CF_CMD_CANCEL_CHAN_ERR_EID (155)
CF Cancel Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Cancel command received without a matching transaction
Definition at line 1419 of file cf_events.h.

10.1.2.51 CF_CMD_CANCEL_INF_EID #define CF_CMD_CANCEL_INF_EID (122)
CF Cancel Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of cancel command
Definition at line 1056 of file cf_events.h.

10.1.2.52 CF_CMD_CHAN_PARAM_ERR_EID #define CF_CMD_CHAN_PARAM_ERR_EID (130)
CF Command Channel Invalid Event ID.

Type: ERROR

Cause:

Command received with channel parameter out of range
Definition at line 1144 of file cf_events.h.

10.1.2.53 CF_CMD_DISABLE_DEQUEUE_ERR_EID #define CF_CMD_DISABLE_DEQUEUE_ERR_EID (162)
CF Disable Dequeue Command Failed Event ID.

Type: ERROR

Cause:

Disable dequeue command was unsuccessful
Definition at line 1496 of file cf_events.h.

10.1.2.54 CF_CMD_DISABLE_DEQUEUE_INF_EID #define CF_CMD_DISABLE_DEQUEUE_INF_EID (125)
CF Disable Dequeue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable dequeue command
Definition at line 1089 of file cf_events.h.

10.1.2.55 CF_CMD_DISABLE_ENGINE_INF_EID #define CF_CMD_DISABLE_ENGINE_INF_EID (117)
CF Disable Engine Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable engine command
Definition at line 1001 of file cf_events.h.

10.1.2.56 CF_CMD_DISABLE_POLLDIR_ERR_EID #define CF_CMD_DISABLE_POLLDIR_ERR_EID (164)
CF Disable Polldir Command Failed Event ID.

Type: ERROR

Cause:

Disable polldir command was unsuccessful
Definition at line 1518 of file cf_events.h.

10.1.2.57 CF_CMD_DISABLE_POLLDIR_INF_EID #define CF_CMD_DISABLE_POLLDIR_INF_EID (127)
CF Disable Polldir Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable polldir command
Definition at line 1111 of file cf_events.h.

10.1.2.58 CF_CMD_ENABLE_DEQUEUE_ERR_EID #define CF_CMD_ENABLE_DEQUEUE_ERR_EID (161)
CF Enable Dequeue Command Failed Event ID.

Type: ERROR

Cause:

Enable Dequeue command was unsuccessful
Definition at line 1485 of file cf_events.h.

10.1.2.59 CF_CMD_ENABLE_DEQUEUE_INF_EID #define CF_CMD_ENABLE_DEQUEUE_INF_EID (124)
CF Enable Dequeue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable dequeue command
Definition at line 1078 of file cf_events.h.

10.1.2.60 CF_CMD_ENABLE_ENGINE_ERR_EID #define CF_CMD_ENABLE_ENGINE_ERR_EID (148)
CF Enable Engine Command Failed Event ID.

Type: ERROR

Cause:

Failed to initialize engine when processing engine enable command
Definition at line 1342 of file cf_events.h.

10.1.2.61 CF_CMD_ENABLE_ENGINE_INF_EID #define CF_CMD_ENABLE_ENGINE_INF_EID (116)
CF Enable Engine Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable engine command
Definition at line 990 of file cf_events.h.

10.1.2.62 CF_CMD_ENABLE_POLLDIR_ERR_EID #define CF_CMD_ENABLE_POLLDIR_ERR_EID (163)
CF Enable Polldir Command Failed Event ID.

Type: ERROR

Cause:

Enable polldir command was unsuccessful
Definition at line 1507 of file cf_events.h.

10.1.2.63 CF_CMD_ENABLE_POLLDIR_INF_EID #define CF_CMD_ENABLE_POLLDIR_INF_EID (126)
CF Enable Polldir Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable polldir command
Definition at line 1100 of file cf_events.h.

10.1.2.64 CF_CMD_ENG_ALREADY_DIS_INF_EID #define CF_CMD_ENG_ALREADY_DIS_INF_EID (150)
CF Disable Engine Command Engine Already Disabled Event ID.

Type: INFORMATION

Cause:

Disable engine command received while engine is already disabled
Definition at line 1364 of file cf_events.h.

10.1.2.65 CF_CMD_ENG_ALREADY_ENA_INF_EID #define CF_CMD_ENG_ALREADY_ENA_INF_EID (149)
CF Enable Engine Command Engine Already Enabled Event ID.

Type: INFORMATION

Cause:

Enable engine command received while engine is already enabled
Definition at line 1353 of file cf_events.h.

10.1.2.66 CF_CMD_FREEZE_ERR_EID #define CF_CMD_FREEZE_ERR_EID (159)
CF Freeze Command Failed Event ID.

Type: ERROR

Cause:

Freeze command was unsuccessful
Definition at line 1463 of file cf_events.h.

10.1.2.67 CF_CMD_FREEZE_INF_EID #define CF_CMD_FREEZE_INF_EID (120)
CF Freeze Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of freeze command
Definition at line 1034 of file cf_events.h.

10.1.2.68 CF_CMD_GETSET1_INF_EID #define CF_CMD_GETSET1_INF_EID (112)
CF Set Parameter Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of set parameter command
Definition at line 946 of file cf_events.h.

10.1.2.69 CF_CMD_GETSET2_INF_EID #define CF_CMD_GETSET2_INF_EID (113)
CF Get Parameter Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of get parameter command
Definition at line 957 of file cf_events.h.

10.1.2.70 CF_CMD_GETSET_CHAN_ERR_EID #define CF_CMD_GETSET_CHAN_ERR_EID (147)
CF Set/Get Parameter Command Invalid Channel Event ID.

Type: ERROR

Cause:

Invalid channel value received in set or get parameter command
Definition at line 1331 of file cf_events.h.

10.1.2.71 CF_CMD_GETSET_PARAM_ERR_EID #define CF_CMD_GETSET_PARAM_ERR_EID (146)
CF Set/Get Parameter Command Invalid Parameter ID Event ID.

Type: ERROR

Cause:

Invalid parameter id value received in set or get parameter command
Definition at line 1320 of file cf_events.h.

10.1.2.72 CF_CMD_GETSET_VALIDATE_ERR_EID #define CF_CMD_GETSET_VALIDATE_ERR_EID (145)
CF Set Parameter Command Parameter Validation Failed Event ID.

Type: ERROR

Cause:

Parameter validation failed during processing of set parameter command
Definition at line 1309 of file cf_events.h.

10.1.2.73 CF_CMD_LEN_ERR_EID #define CF_CMD_LEN_ERR_EID (151)
CF Command Length Verification Failed Event ID.

Type: ERROR

Cause:

Received command length verification failure
Definition at line 1375 of file cf_events.h.

10.1.2.74 CF_CMD_PLAYBACK_DIR_ERR_EID #define CF_CMD_PLAYBACK_DIR_ERR_EID (158)
CF Playback Directory Command Failed Event ID.

Type: ERROR

Cause:

Playback directory command was unsuccessful
Definition at line 1452 of file cf_events.h.

10.1.2.75 CF_CMD_PLAYBACK_DIR_INF_EID #define CF_CMD_PLAYBACK_DIR_INF_EID (119)
CF Playback Directory Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of playback directory command
Definition at line 1023 of file cf_events.h.

10.1.2.76 CF_CMD_POLLDIR_INVALID_ERR_EID #define CF_CMD_POLLDIR_INVALID_ERR_EID (135)
CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.

Type: ERROR

Cause:

Enable/disable polling directory command received with invalid poling directory index
Definition at line 1199 of file cf_events.h.

10.1.2.77 CF_CMD_PURGE_ARG_ERR_EID #define CF_CMD_PURGE_ARG_ERR_EID (136)
CF Purge Queue Command Invalid Argument Event ID.

Type: ERROR

Cause:

Purge Queue command received with invalid queue argument
Definition at line 1210 of file cf_events.h.

10.1.2.78 CF_CMD_PURGE_QUEUE_ERR_EID #define CF_CMD_PURGE_QUEUE_ERR_EID (165)
CF Purge Queue Command Failed Event ID.

Type: ERROR

Cause:

Purge queue command was unsuccessful
Definition at line 1529 of file cf_events.h.

10.1.2.79 CF_CMD_PURGE_QUEUE_INF_EID #define CF_CMD_PURGE_QUEUE_INF_EID (128)
CF Purge Queue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of purge queue command
Definition at line 1122 of file cf_events.h.

10.1.2.80 CF_CMD_RESET_INVALID_ERR_EID #define CF_CMD_RESET_INVALID_ERR_EID (129)
CF Reset Counters Command Invalid Event ID.

Type: ERROR

Cause:

Reset counters command received with invalid parameter
Definition at line 1133 of file cf_events.h.

10.1.2.81 CF_CMD_SUSPRES_CHAN_ERR_EID #define CF_CMD_SUSPRES_CHAN_ERR_EID (134)
CF Suspend/Resume Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Suspend/resume command received without a matching transaction
Definition at line 1188 of file cf_events.h.

10.1.2.82 CF_CMD_SUSPRES_INF_EID #define CF_CMD_SUSPRES_INF_EID (114)
CF Suspend/Resume Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of suspend/resume command
Definition at line 968 of file cf_events.h.

10.1.2.83 CF_CMD_SUSPRES_SAME_INF_EID #define CF_CMD_SUSPRES_SAME_INF_EID (133)
CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.

Type: INFORMATION

Cause:

Suspend/resume command received affecting single transaction already set to that state
Definition at line 1177 of file cf_events.h.

10.1.2.84 CF_CMD_THAW_ERR_EID #define CF_CMD_THAW_ERR_EID (160)
CF Thaw Command Failed Event ID.

Type: ERROR

Cause:

Thaw command was unsuccessful
Definition at line 1474 of file cf_events.h.

10.1.2.85 CF_CMD_THAW_INF_EID #define CF_CMD_THAW_INF_EID (121)
CF Thaw Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of thaw command
Definition at line 1045 of file cf_events.h.

10.1.2.86 CF_CMD_TRANS_NOT_FOUND_ERR_EID #define CF_CMD_TRANS_NOT_FOUND_ERR_EID (131)
CF Command Transaction Invalid Event ID.

Type: ERROR

Cause:

Command received without a matching transaction
Definition at line 1155 of file cf_events.h.

10.1.2.87 CF_CMD_TSN_CHAN_INVALID_ERR_EID #define CF_CMD_TSN_CHAN_INVALID_ERR_EID (132)
CF Command All Transaction Channel Invalid Event ID.

Type: ERROR

Cause:

Command received to act on all transactions with invalid channel
Definition at line 1166 of file cf_events.h.

10.1.2.88 CF_CMD_TX_FILE_ERR_EID #define CF_CMD_TX_FILE_ERR_EID (157)
CF Transfer File Command Failed Event ID.

Type: ERROR

Cause:

Transfer file command was unsuccessful
Definition at line 1441 of file cf_events.h.

10.1.2.89 CF_CMD_TX_FILE_INF_EID #define CF_CMD_TX_FILE_INF_EID (118)
CF Transfer File Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of transfer file command
Definition at line 1012 of file cf_events.h.

10.1.2.90 CF_CMD_WHIST_WRITE_ERR_EID #define CF_CMD_WHIST_WRITE_ERR_EID (153)
CF Write Entry To File Failed Event ID.

Type: ERROR

Cause:

Write entry to file did not match expected length
Definition at line 1397 of file cf_events.h.

10.1.2.91 CF_CMD_WQ_ARGS_ERR_EID #define CF_CMD_WQ_ARGS_ERR_EID (138)
CF Write Queue Command Invalid Queue Event ID.

Type: ERROR

Cause:

Write Queue command received with invalid queue selection arguments
Definition at line 1232 of file cf_events.h.

10.1.2.92 CF_CMD_WQ_CHAN_ERR_EID #define CF_CMD_WQ_CHAN_ERR_EID (137)
CF Write Queue Command Invalid Channel Event ID.

Type: ERROR

Cause:

Write Queue command received with invalid channel argument
Definition at line 1221 of file cf_events.h.

10.1.2.93 CF_CMD_WQ_INF_EID #define CF_CMD_WQ_INF_EID (115)
CF Write Queue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of write queue command
Definition at line 979 of file cf_events.h.

10.1.2.94 CF_CMD_WQ_OPEN_ERR_EID #define CF_CMD_WQ_OPEN_ERR_EID (139)
CF Write Queue Command File Open Failed Event ID.

Type: ERROR

Cause:

Failure of open file call during processing of write queue command
Definition at line 1243 of file cf_events.h.

10.1.2.95 CF_CMD_WQ_WRITEHIST_RX_ERR_EID #define CF_CMD_WQ_WRITEHIST_RX_ERR_EID (141)
CF Write Queue Command RX History File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for RX history during processing of write queue command
Definition at line 1265 of file cf_events.h.

10.1.2.96 CF_CMD_WQ_WRITEHIST_TX_ERR_EID #define CF_CMD_WQ_WRITEHIST_TX_ERR_EID (144)
CF Write Queue Command TX History File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX history during processing of write queue command
Definition at line 1298 of file cf_events.h.

10.1.2.97 CF_CMD_WQ_WRITEQ_PEND_ERR_EID #define CF_CMD_WQ_WRITEQ_PEND_ERR_EID (143)
CF Write Queue Command TX Pending File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX pending transactions during processing of write queue command
Definition at line 1287 of file cf_events.h.

10.1.2.98 CF_CMD_WQ_WRITEQ_RX_ERR_EID #define CF_CMD_WQ_WRITEQ_RX_ERR_EID (140)
CF Write Queue Command RX Active File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for RX active transactions during processing of write queue command
Definition at line 1254 of file cf_events.h.

10.1.2.99 CF_CMD_WQ_WRITEQ_TX_ERR_EID #define CF_CMD_WQ_WRITEQ_TX_ERR_EID (142)
CF Write Queue Command TX Active File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX active transactions during processing of write queue command
Definition at line 1276 of file cf_events.h.

10.1.2.100 CF_CR_CHANNEL_PIPE_ERR_EID #define CF_CR_CHANNEL_PIPE_ERR_EID (31)
CF Channel Create Pipe Failed Event ID.

Type: ERROR

Cause:

Failure from create pipe call during engine channel initialization
Definition at line 169 of file cf_events.h.

10.1.2.101 CF_CR_PIPE_ERR_EID #define CF_CR_PIPE_ERR_EID (36)
CF Create SB Command Pipe at Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from create command pipe call during application initialization
Definition at line 224 of file cf_events.h.

10.1.2.102 CF_INIT_CRC_ALIGN_ERR_EID #define CF_INIT_CRC_ALIGN_ERR_EID (34)
CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.

Type: ERROR

Cause:

Configuration table CRC bytes per wakeup not aligned or zero
Definition at line 202 of file cf_events.h.

10.1.2.103 CF_INIT_INF_EID #define CF_INIT_INF_EID (20)
CF Initialization Event ID.

Type: INFORMATION

Cause:

Successful completion of application initialization
Definition at line 47 of file cf_events.h.

10.1.2.104 CF_INIT_MSG_RECV_ERR_EID #define CF_INIT_MSG_RECV_ERR_EID (29)
CF SB Receive Buffer Failed Event ID.

Type: ERROR

Cause:

Failure from SB Receive Buffer call in application run loop
Definition at line 146 of file cf_events.h.

10.1.2.105 CF_INIT_OUTGOING_SIZE_ERR_EID #define CF_INIT_OUTGOING_SIZE_ERR_EID (35)
CF Outgoing Chunk Size Config Table Validation Failed Event ID.

Type: ERROR

Cause:

Configuration table outgoing chunk size larger than PDU data size
Definition at line 213 of file cf_events.h.

10.1.2.106 CF_INIT_SEM_ERR_EID #define CF_INIT_SEM_ERR_EID (30)
CF Channel Semaphore Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from get semaphore by name call during engine channel initialization, semaphore needs to exist before engine is initialized.
Definition at line 158 of file cf_events.h.

10.1.2.107 CF_INIT_SUB_ERR_EID #define CF_INIT_SUB_ERR_EID (32)
CF Channel Message Subscription Failed Event ID.

Type: ERROR

Cause:

Failure from message subscription call during engine channel initialization
Definition at line 180 of file cf_events.h.

10.1.2.108 CF_INIT_TBL_CHECK_GA_ERR_EID #define CF_INIT_TBL_CHECK_GA_ERR_EID (23)
CF Check Table Get Address Failed Event ID.

Type: ERROR

Cause:

Failure from get table call during periodic table check
Definition at line 80 of file cf_events.h.

10.1.2.109 CF_INIT_TBL_CHECK_MAN_ERR_EID #define CF_INIT_TBL_CHECK_MAN_ERR_EID (22)
CF Check Table Manage Failed Event ID.

Type: ERROR

Cause:

Failure from manage table call during periodic table check
Definition at line 69 of file cf_events.h.

10.1.2.110 CF_INIT_TBL_CHECK_REL_ERR_EID #define CF_INIT_TBL_CHECK_REL_ERR_EID (21)
CF Check Table Release Address Failed Event ID.

Type: ERROR

Cause:

Failure from release table address call during periodic table check
Definition at line 58 of file cf_events.h.

10.1.2.111 CF_INIT_TBL_GETADDR_ERR_EID #define CF_INIT_TBL_GETADDR_ERR_EID (27)
CF Table Get Address At Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from table get address call during application initialization
Definition at line 124 of file cf_events.h.

10.1.2.112 CF_INIT_TBL_LOAD_ERR_EID #define CF_INIT_TBL_LOAD_ERR_EID (25)
CF Table Load At Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from table load call during application initialization
Definition at line 102 of file cf_events.h.

10.1.2.113 CF_INIT_TBL_MANAGE_ERR_EID #define CF_INIT_TBL_MANAGE_ERR_EID (26)
CF Table Manage At Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from table manage call during application initialization
Definition at line 113 of file cf_events.h.

10.1.2.114 CF_INIT_TBL_REG_ERR_EID #define CF_INIT_TBL_REG_ERR_EID (24)
CF Table Registration At Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from table register call during application initialization
Definition at line 91 of file cf_events.h.

10.1.2.115 CF_INIT_TPS_ERR_EID #define CF_INIT_TPS_ERR_EID (33)
CF Ticks Per Second Config Table Validation Failed Event ID.

Type: ERROR

Cause:

Configuration table ticks per second set to zero
Definition at line 191 of file cf_events.h.

10.1.2.116 CF_MID_ERR_EID #define CF_MID_ERR_EID (28)
CF Message ID Invalid Event ID.

Type: ERROR

Cause:

Invalid message ID received on the software bus pipe
Definition at line 135 of file cf_events.h.

10.1.2.117 CF_NOOP_INF_EID #define CF_NOOP_INF_EID (110)
CF NOOP Command Received Event ID.

Type: INFORMATION

Cause:

Receipt of NOOP command
Definition at line 924 of file cf_events.h.

10.1.2.118 CF_PDU_ACK_SHORT_ERR_EID #define CF_PDU_ACK_SHORT_ERR_EID (48)
CF Acknowledgment PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing acknowledgment PDU
Definition at line 316 of file cf_events.h.

10.1.2.119 CF_PDU_EOF_SHORT_ERR_EID #define CF_PDU_EOF_SHORT_ERR_EID (47)
CF End-Of-File PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing end-of-file PDU
Definition at line 305 of file cf_events.h.

10.1.2.120 CF_PDU_FD_SHORT_ERR_EID #define CF_PDU_FD_SHORT_ERR_EID (46)
CF File Data PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing file data PDU
Definition at line 294 of file cf_events.h.

10.1.2.121 CF_PDU_FD_UNSUPPORTED_ERR_EID #define CF_PDU_FD_UNSUPPORTED_ERR_EID (54)
CF File Data PDU Unsupported Option Event ID.

Type: ERROR

Cause:

File Data PDU received with the segment metadata flag set
Definition at line 349 of file cf_events.h.

10.1.2.122 CF_PDU_FIN_SHORT_ERR_EID #define CF_PDU_FIN_SHORT_ERR_EID (49)
CF Finished PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing finished PDU
Definition at line 327 of file cf_events.h.

10.1.2.123 CF_PDU_INVALID_DST_LEN_ERR_EID #define CF_PDU_INVALID_DST_LEN_ERR_EID (45)
CF Metadata PDU Destination Filename Length Invalid Event ID.

Type: ERROR

Cause:

Metadata PDU destination filename length exceeds buffer size
Definition at line 283 of file cf_events.h.

10.1.2.124 CF_PDU_INVALID_SRC_LEN_ERR_EID #define CF_PDU_INVALID_SRC_LEN_ERR_EID (44)
CF Metadata PDU Source Filename Length Invalid Event ID.

Type: ERROR

Cause:

Metadata PDU source filename length exceeds buffer size
Definition at line 272 of file cf_events.h.

10.1.2.125 CF_PDU_LARGE_FILE_ERR_EID #define CF_PDU_LARGE_FILE_ERR_EID (55)
CF PDU Header Large File Flag Set Event ID.

Type: ERROR

Cause:

PDU Header received with the unsupported large file flag set
Definition at line 360 of file cf_events.h.

10.1.2.126 CF_PDU_MD_RECVD_INF_EID #define CF_PDU_MD_RECVD_INF_EID (40)
CF Metadata PDU Received Event ID.

Type: INFORMATION

Cause:

Successful processing of metadata PDU
Definition at line 239 of file cf_events.h.

10.1.2.127 CF_PDU_MD_SHORT_ERR_EID #define CF_PDU_MD_SHORT_ERR_EID (43)
CF Metadata PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing metadata PDU
Definition at line 261 of file cf_events.h.

10.1.2.128 CF_PDU_NAK_SHORT_ERR_EID #define CF_PDU_NAK_SHORT_ERR_EID (50)
CF Negative Acknowledgment PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing negative acknowledgment PDU
Definition at line 338 of file cf_events.h.

10.1.2.129 CF_PDU_SHORT_HEADER_ERR_EID #define CF_PDU_SHORT_HEADER_ERR_EID (41)
CF PDU Header Too Short Event ID.

Type: ERROR

Cause:

Failure processing PDU header
Definition at line 250 of file cf_events.h.

10.1.2.130 CF_PDU_TRUNCATION_ERR_EID #define CF_PDU_TRUNCATION_ERR_EID (56)
CF PDU Header Field Truncation.

Type: ERROR

Cause:

PDU Header received with fields that would be truncated with the cf configuration
Definition at line 371 of file cf_events.h.

10.1.2.131 CF_RESET_FREED_XACT_DBG_EID #define CF_RESET_FREED_XACT_DBG_EID (59)
Attempt to reset a transaction that has already been freed.

Type: DEBUG

Cause:

Can be induced via various off-nominal conditions - such as sending a META-data PDU with an invalid file destination.
Definition at line 388 of file cf_events.h.

10.1.2.132 CF_RESET_INF_EID #define CF_RESET_INF_EID (111)
CF Reset Counters Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of reset counters command
Definition at line 935 of file cf_events.h.

10.2 CFS CFDP Mission Configuration

Macros

- `#define CF_PERF_ID_APPMAIN (11)`
Main application performance ID.
- `#define CF_PERF_ID_FSEEK (12)`
File seek performance ID.
- `#define CF_PERF_ID_FOPEN (13)`
File open performance ID.
- `#define CF_PERF_ID_FCLOSE (14)`
File close performance ID.
- `#define CF_PERF_ID_FREAD (15)`
File read performance ID.
- `#define CF_PERF_ID_FWRITE (16)`
File write performance ID.
- `#define CF_PERF_ID_CYCLE_ENG (17)`
Cycle engine performance ID.
- `#define CF_PERF_ID_DIRREAD (18)`
Directory read performance ID.
- `#define CF_PERF_ID_CREAT (19)`
Create performance ID.
- `#define CF_PERF_ID_RENAME (20)`
Rename performance ID.
- `#define CF_PERF_ID_PDURCVD(x) (30 + x)`
PDU Received performance ID.
- `#define CF_PERF_ID_PDUSENT(x) (40 + x)`
PDU Sent performance ID.

10.2.1 Detailed Description

10.2.2 Macro Definition Documentation

10.2.2.1 CF_PERF_ID_APPMAIN `#define CF_PERF_ID_APPMAIN (11)`

Main application performance ID.

Definition at line 33 of file cf_perfids.h.

10.2.2.2 CF_PERF_ID_CREAT `#define CF_PERF_ID_CREAT (19)`

Create performance ID.

Definition at line 41 of file cf_perfids.h.

10.2.2.3 CF_PERF_ID_CYCLE_ENG `#define CF_PERF_ID_CYCLE_ENG (17)`

Cycle engine performance ID.

Definition at line 39 of file cf_perfids.h.

10.2.2.4 CF_PERF_ID_DIRREAD #define CF_PERF_ID_DIRREAD (18)
Directory read performance ID.
Definition at line 40 of file cf_perfids.h.

10.2.2.5 CF_PERF_ID_FCLOSE #define CF_PERF_ID_FCLOSE (14)
File close performance ID.
Definition at line 36 of file cf_perfids.h.

10.2.2.6 CF_PERF_ID_FOPEN #define CF_PERF_ID_FOPEN (13)
File open performance ID.
Definition at line 35 of file cf_perfids.h.

10.2.2.7 CF_PERF_ID_FREAD #define CF_PERF_ID_FREAD (15)
File read performance ID.
Definition at line 37 of file cf_perfids.h.

10.2.2.8 CF_PERF_ID_FSEEK #define CF_PERF_ID_FSEEK (12)
File seek performance ID.
Definition at line 34 of file cf_perfids.h.

10.2.2.9 CF_PERF_ID_FWRITE #define CF_PERF_ID_FWRITE (16)
File write performance ID.
Definition at line 38 of file cf_perfids.h.

10.2.2.10 CF_PERF_ID_PDURCVD #define CF_PERF_ID_PDURCVD (x) (30 + x)
PDU Received performance ID.
Definition at line 44 of file cf_perfids.h.

10.2.2.11 CF_PERF_ID_PDUSENT #define CF_PERF_ID_PDUSENT (x) (40 + x)
PDU Sent performance ID.
Definition at line 45 of file cf_perfids.h.

10.2.2.12 CF_PERF_ID_RENAME #define CF_PERF_ID_RENAME (20)
Rename performance ID.
Definition at line 42 of file cf_perfids.h.

10.3 CFS CFDP Version

Version Numbers

Macros

- `#define CF_MAJOR_VERSION (3)`
Major version number.
- `#define CF_MINOR_VERSION (0)`
Minor version number.
- `#define CF_REVISION (99)`
Revision number.

10.3.1 Detailed Description

Version Numbers

10.3.2 Macro Definition Documentation

10.3.2.1 CF_MAJOR_VERSION `#define CF_MAJOR_VERSION (3)`

Major version number.

Definition at line 35 of file cf_version.h.

10.3.2.2 CF_MINOR_VERSION `#define CF_MINOR_VERSION (0)`

Minor version number.

Definition at line 36 of file cf_version.h.

10.3.2.3 CF_REVISION `#define CF_REVISION (99)`

Revision number.

Definition at line 37 of file cf_version.h.

10.4 CFS CFDP Command Codes

Enumerations

- enum `CF_CMDS` {
 `CF_NOOP_CC` = 0, `CF_RESET_CC` = 1, `CF_TX_FILE_CC` = 2, `CF_PLAYBACK_DIR_CC` = 3,
 `CF_FREEZE_CC` = 4, `CF_THAW_CC` = 5, `CF_SUSPEND_CC` = 6, `CF_RESUME_CC` = 7,
 `CF_CANCEL_CC` = 8, `CF_ABANDON_CC` = 9, `CF_SET_PARAM_CC` = 10, `CF_GET_PARAM_CC` = 11,
 `CF_WRITE_QUEUE_CC` = 15, `CF_ENABLE_DEQUEUE_CC` = 16, `CF_DISABLE_DEQUEUE_CC` = 17,
 `CF_ENABLE_DIR_POLLING_CC` = 18,
 `CF_DISABLE_DIR_POLLING_CC` = 19, `CF_PURGE_QUEUE_CC` = 21, `CF_ENABLE_ENGINE_CC` = 22,
 `CF_DISABLE_ENGINE_CC` = 23,
 `CF_NUM_COMMANDS` = 24 }

10.4.1 Detailed Description

10.4.2 Enumeration Type Documentation

10.4.2.1 `CF_CMDS` enum `CF_CMDS`

Enumerator

CF_NOOP_CC	<p>No-op.</p> <p>Description</p> <p>No-operation command for aliveness verification and version reporting</p> <p>Command Structure</p> <p>No Payload / Arguments</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_NOOP_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p>
------------	---

Enumerator

CF_RESET_CC	<p>Reset counters.</p> <p>Description</p> <p>Resets the requested housekeeping counters</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t where byte[0] specifies the counters type, byte[1-3] don't care:</p> <ul style="list-style-type: none"> • 0 = all counters • 1 = command counters • 2 = fault counters • 3 = up counters • 4 = down counters <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_RESET_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid counter type, CF_CMD_RESET_INVALID_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p>
-------------	--

Enumerator

CF_TX_FILE_CC	<p>Transmit file.</p> <p>Description</p> <p>Requests transmission of a file</p> <p>Command Structure</p> <p>CF_TxFileCmd_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_TX_FILE_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid parameter, CF_CMD_BAD_PARAM_ERR_EID • Transaction initialization failure, CF_CMD_TX_FILE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_PLAYBACK_DIR_CC</p>
----------------------	--

Enumerator

CF_PLAYBACK_DIR_CC	<p>Playback a directory.</p> <p>Description</p> <p>Transmits all the files in a directory</p> <p>Command Structure</p> <p>CF_PlaybackDirCmd_t - note it's currently a typedef of CF_TxFileCmd_t, where the source filename and destination filename are directories</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_PLAYBACK_DIR_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid parameter, CF_CMD_BAD_PARAM_ERR_EID • Playback initialization failure, CF_CMD_PLAYBACK_DIR_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_TX_FILE_CC</p>
---------------------------	---

Enumerator

CF_FREEZE_CC	<p>Freeze a channel.</p> <p>Description</p> <p>Disables the transmission of all PDUs and disables tick processing (timeouts, ACK/NAK, etc) for the specified channel, will still consume all received messages. Note this could cause failures for class 2 transactions in progress.</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_FREEZE_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Command processing failure, CF_CMD_FREEZE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_THAW_CC</p>
---------------------	--

Enumerator

CF_THAW_CC	<p>Thaw a channel.</p> <p>Description</p> <p>Enables the transmission of all PDUs and resumes tick processing (timeouts, ACK/NAK, etc) for the specified channel, note received messages are consumed either way.</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_THAW_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Command processing failure, CF_CMD_THAW_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_FREEZE_CC</p>
------------	---

Enumerator

CF_SUSPEND_CC	<p>Suspend a transaction.</p> <p>Description</p> <p>Disables the transmission of all PDUs and disables tick processing (timeouts, ACK/NAK, etc) on a single transaction, all channels and transactions, or all transactions on a specific channel. Will still consume all received messages. Note suspension is tracked per transaction, whereas freeze/thaw are tracked per channel.</p> <p>Command Structure</p> <p>CF_Transaction_Payload_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_SUSPRES_INF_EID • CF_CMD_SUSPRES_SAME_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Transaction not found using compound key, CF_CMD_TRANS_NOT_FOUND_ERR_EID • Invalid channel number, CF_CMD_TSN_CHAN_INVALID_ERR_EID • No matching transaction, CF_CMD_SUSPRES_CHAN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_RESUME_CC, CF_CANCEL_CC, CF_ABANDON_CC</p>
----------------------	--

Enumerator

CF_RESUME_CC	<p>Resume a transaction.</p> <p>Description</p> <p>Enables the transmission of all PDUs and resumes tick processing (timeouts, ACK/NAK, etc) on a single transaction, all channels and transactions, or all transactions on a specific channel. Note a suspended transaction still consume all received messages. Note suspension is tracked per transaction, whereas freeze/thaw are tracked per channel.</p> <p>Command Structure</p> <p>CF_Transaction_Payload_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_SUSPRES_INF_EID • CF_CMD_SUSPRES_SAME_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Transaction not found using compound key, CF_CMD_TRANS_NOT_FOUND_ERR_EID • Invalid channel number, CF_CMD_TSN_CHAN_INVALID_ERR_EID • No matching transaction, CF_CMD_SUSPRES_CHAN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_SUSPEND_CC, CF_CANCEL_CC, CF_ABANDON_CC</p>
---------------------	--

Enumerator

CF_CANCEL_CC	<p>Cancel a transaction.</p> <p>Description</p> <p>Cancel transaction processing by taking steps to close out cleanly (based on transaction type and direction) for a single transaction, all channels and transactions, or all transactions on a specific channel.</p> <p>Command Structure</p> <p>CF_Transaction_Payload_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_CANCEL_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Transaction not found using compound key, CF_CMD_TRANS_NOT_FOUND_ERR_EID • Invalid channel number, CF_CMD_TSN_CHAN_INVALID_ERR_EID • No matching transaction, CF_CMD_CANCEL_CHAN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_SUSPEND_CC, CF_RESUME_CC, CF_ABANDON_CC</p>
--------------	--

Enumerator

CF_ABANDON_CC	<p>Abandon a transaction.</p> <p>Description</p> <p>Abandon transaction processing with an immediate reset (no close out attempted) for a single transaction, all channels and transactions, or all transactions on a specific channel.</p> <p>Command Structure</p> <p>CF_Transaction_Payload_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_ABANDON_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Transaction not found using compound key, CF_CMD_TRANS_NOT_FOUND_ERR_EID • Invalid channel number, CF_CMD_TSN_CHAN_INVALID_ERR_EID • No matching transaction, CF_CMD_ABANDON_CHAN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_SUSPEND_CC, CF_RESUME_CC, CF_CANCEL_CC</p>
---------------	---

Enumerator

CF_SET_PARAM_CC	<p>Set parameter.</p> <p>Description</p> <p>Sets a configuration parameter</p> <p>Command Structure</p> <p>CF_SetParamCmd_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_GETSET1_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid configuration parameter key, CF_CMD_GETSET_PARAM_ERR_EID • Invalid channel number, CF_CMD_GETSET_CHAN_ERR_EID • Parameter value failed validation, CF_CMD_GETSET_VALIDATE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_GET_PARAM_CC</p>
------------------------	--

Enumerator

CF_GET_PARAM_CC	<p>Get parameter.</p> <p>Description</p> <p>Gets a configuration parameter</p> <p>Command Structure</p> <p>CF_GetParamCmd_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_GETSET2_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid configuration parameter key, CF_CMD_GETSET_PARAM_ERR_EID • Invalid channel number, CF_CMD_GETSET_CHAN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_SET_PARAM_CC</p>
------------------------	---

Enumerator

CF_WRITE_QUEUE_CC	<p>Write queue.</p> <p>Description</p> <p>Writes requested queue(s) to a file</p> <p>Command Structure</p> <p>CF_WriteQueueCmd_t</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_WQ_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid parameter combination, CF_CMD_WQ_ARGS_ERR_EID • Invalid channel number, CF_CMD_WQ_CHAN_ERR_EID • Open file to write failed, CF_CMD_WQ_OPEN_ERR_EID • Write RX data failed, CF_CMD_WQ_WRITEQ_RX_ERR_EID • Write RX history data failed, CF_CMD_WQ_WRITEHIST_RX_ERR_EID • Write TX data failed, CF_CMD_WQ_WRITEQ_TX_ERR_EID • Write TX history data failed, CF_CMD_WQ_WRITEHIST_TX_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_PURGE_QUEUE_CC</p>
--------------------------	---

Enumerator

CF_ENABLE_DEQUEUE_CC	<p>Description</p> <p>Enables the sending of file data PDUs.</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_ENABLE_DEQUEUE_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Enable dequeue failed, CF_CMD_ENABLE_DEQUEUE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_DISABLE_DEQUEUE_CC</p>
-----------------------------	--

Enumerator

CF_DISABLE_DEQUEUE_CC	<p>Disable dequeue.</p> <p>Description</p> <p>Disables the sending of file data PDUs.</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t where byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_DISABLE_DEQUEUE_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Disable dequeue failed, CF_CMD_DISABLE_DEQUEUE_INF_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_ENABLE_DEQUEUE_CC</p>
------------------------------	---

Enumerator

CF_ENABLE_DIR_POLLING_CC	<p>Enable directory polling.</p> <p>Description</p> <p>Enables the processing of polling directories</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t</p> <p>byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>byte[1] specifies the polling directory index</p> <ul style="list-style-type: none"> • 255 = all polling directories • else = single polling directory index <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_ENABLE_POLLDIR_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Invalid polling directory index, CF_CMD_POLLDIR_INVALID_ERR_EID • Enable directory polling failed, CF_CMD_ENABLE_POLLDIR_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_DISABLE_DIR_POLLING_CC</p>
---------------------------------	--

Enumerator

	<p>CF_DISABLE_DIR_POLLING_CC</p> <p>Disable directory polling.</p> <p>Description</p> <p>Disable the processing of polling directories</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t</p> <p>byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>byte[1] specifies the polling directory index</p> <ul style="list-style-type: none"> • 255 = all polling directories • else = single polling directory index <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_DISABLE_POLLDIR_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Invalid polling directory index, CF_CMD_POLLDIR_INVALID_ERR_EID • Disable directory polling failed, CF_CMD_DISABLE_POLLDIR_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_ENABLE_DIR_POLLING_CC</p>
--	---

Enumerator

CF_PURGE_QUEUE_CC	<p>Purge queue.</p> <p>Description</p> <p>Purge the requested queue</p> <p>Command Structure</p> <p>CF_UnionArgs_Payload_t</p> <p>byte[0] specifies the channel number or all channels</p> <ul style="list-style-type: none"> • 255 = all channels • else = single channel <p>byte[1] specifies the queue</p> <ul style="list-style-type: none"> • 0 = Pending queue • 1 = History queue • 2 = Both pending and history queue <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_PURGE_QUEUE_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Invalid channel number, CF_CMD_CHAN_PARAM_ERR_EID • Invalid purge queue argument, CF_CMD_PURGE_ARG_ERR_EID • Purge queue failed, CF_CMD_PURGE_QUEUE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_WRITE_QUEUE_CC</p>
--------------------------	---

Enumerator

CF_ENABLE_ENGINE_CC	<p>Enable engine.</p> <p>Description</p> <p>Reinitialize engine and enable processing. Note configuration table updates are not processed while the engine is enabled.</p> <p>Command Structure</p> <p>No Payload / Arguments</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_ENABLE_ENGINE_INF_EID • CF_CMD_ENG_ALREADY_ENA_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID • Engine initialization failed, CF_CMD_ENABLE_ENGINE_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_DISABLE_ENGINE_CC</p>
----------------------------	--

Enumerator

CF_DISABLE_ENGINE_CC	<p>Disable engine.</p> <p>Description</p> <p>Disable engine processing. Note configuration table updates can be performed while the engine is disabled, and when the engine is re-enabled the new configuration will take effect.</p> <p>Command Structure</p> <p>No Payload / Arguments</p> <p>Command Verification</p> <p>Successful execution of this command may be verified with the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd will increment • CF_CMD_DISABLE_ENGINE_INF_EID • CF_CMD_ENG_ALREADY_DIS_INF_EID <p>Error Conditions</p> <p>This command may fail for the following reason(s):</p> <ul style="list-style-type: none"> • Command packet length not as expected, CF_CMD_LEN_ERR_EID <p>Evidence of failure may be found in the following telemetry:</p> <ul style="list-style-type: none"> • CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.terr will increment <p>Criticality</p> <p>None</p> <p>See also</p> <p>CF_DISABLE_ENGINE_CC</p>
CF_NUM_COMMANDS	Command code limit used for validity check and array sizing.

Definition at line 41 of file default_cf_fcncodes.h.

10.5 CFS CFDP Platform Configuration

Macros

- `#define CF_NUM_CHANNELS (2)`
Number of channels.
- `#define CF_NAK_MAX_SEGMENTS (58)`
Max NAK segments supported in a NAK PDU.
- `#define CF_MAX_POLLING_DIR_PER_CHAN (5)`
Max number of polling directories per channel.
- `#define CF_MAX_PDU_SIZE (512)`
Max PDU size.
- `#define CF_FILENAME_MAX_NAME CFE_MISSION_MAX_FILE_LEN`
Maximum file name length.
- `#define CF_FILENAME_MAX_LEN CFE_MISSION_MAX_PATH_LEN`
Max filename and path length.
- `#define CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES 0`
Number of trailing bytes to add to CFDP PDU.
- `#define CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION`
RX chunks per transaction (per channel)
- `#define CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION`
TX chunks per transaction (per channel)
- `#define CF_PIPE_DEPTH (32)`
Application Pipe Depth.
- `#define CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN (10)`
Number of max commanded playback files per chan.
- `#define CF_MAX_SIMULTANEOUS_RX (5)`
Max number of simultaneous file receives.
- `#define CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN (2)`
Max number of commanded playback directories per channel.
- `#define CF_NUM_HISTORIES_PER_CHANNEL (256)`
Number of histories per channel.
- `#define CF_NUM_TRANSACTIONS_PER_PLAYBACK (5)`
Number of transactions per playback directory.
- `#define CF_CONFIG_TABLE_NAME ("config_table")`
Name of the CF Configuration Table.
- `#define CF_CONFIG_TABLE_FILENAME ("/cf/cf_def_config.tbl")`
CF Configuration Table Filename.
- `#define CF_R2_CRC_CHUNK_SIZE (1024)`
R2 CRC calc chunk size.
- `#define CF_RCVMSG_TIMEOUT (100)`
Number of milliseconds to wait for a SB message.
- `#define CF_STARTUP_SEM_MAX_RETRIES 25`
Limits the number of retries to obtain the CF throttle sem.
- `#define CF_STARTUP_SEM_TASK_DELAY 100`
Number of milliseconds to wait if CF throttle sem is not available.

10.5.1 Detailed Description

10.5.2 Macro Definition Documentation

10.5.2.1 CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION `#define CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION`

Value:

```
{  
    CF_NAK_MAX_SEGMENTS, CF_NAK_MAX_SEGMENTS \\\n}
```

RX chunks per transaction (per channel)

Description:

Number of chunks per transaction per channel (RX).

CHUNKS - A chunk is a representation of a range (offset, size) of data received by a receiver.

Class 2 CFDP deals with NAK, so received data must be tracked for receivers in order to generate the NAK. The sender must also keep track of NAK requests and send new file data PDUs as a result. (array size must be CF_NUM_CHANNELS) CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION is an array for each channel indicating the number of chunks per transaction CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION is an array for each channel indicating the number of chunks to keep track of NAK requests from the receiver per transaction

Limits:

Definition at line 78 of file default_cf_internal_cfg.h.

10.5.2.2 CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION `#define CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION`

Value:

```
{  
    CF_NAK_MAX_SEGMENTS, CF_NAK_MAX_SEGMENTS \\\n}
```

TX chunks per transaction (per channel)

Description:

Number of chunks per transaction per channel (TX).

Limits:

Definition at line 92 of file default_cf_internal_cfg.h.

10.5.2.3 CF_CONFIG_TABLE_FILENAME `#define CF_CONFIG_TABLE_FILENAME ("cf/cf_def_config.tbl")`

CF Configuration Table Filename.

Description:

The value of this constant defines the filename of the CF Config Table

Limits

The length of this string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 190 of file default_cf_internal_cfg.h.

10.5.2.4 CF_CONFIG_TABLE_NAME #define CF_CONFIG_TABLE_NAME ("config_table")
Name of the CF Configuration Table.

Description:

This parameter defines the name of the CF Configuration Table.

Limits

The length of this string, including the NULL terminator cannot exceed the [OS_MAX_PATH_LEN](#) value.

Definition at line 178 of file default_cf_internal_cfg.h.

10.5.2.5 CF_FILENAME_MAX_LEN #define CF_FILENAME_MAX_LEN CFE_MISSION_MAX_PATH_LEN
Max filename and path length.

Limits:

Definition at line 116 of file default_cf_interface_cfg.h.

10.5.2.6 CF_FILENAME_MAX_NAME #define CF_FILENAME_MAX_NAME CFE_MISSION_MAX_FILE_LEN
Maximum file name length.

Limits:

Definition at line 108 of file default_cf_interface_cfg.h.

10.5.2.7 CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN #define CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN (2)
Max number of commanded playback directories per channel.

Description:

Each channel can support this number of ground commanded directory playbacks.

Limits:

Definition at line 143 of file default_cf_internal_cfg.h.

10.5.2.8 CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN #define CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN (10)
Number of max commanded playback files per chan.

Description:

This is the max number of outstanding ground commanded file transmits per channel.

Limits:

Definition at line 119 of file default_cf_internal_cfg.h.

10.5.2.9 CF_MAX_PDU_SIZE #define CF_MAX_PDU_SIZE (512)

Max PDU size.

Description:

Limits the maximum possible Tx PDU size. Note the resulting CCSDS packet also includes a CCSDS header and CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES. The outgoing file data chunk size is also limited from the table configuration or by set parameter command, which is checked against this value (+ smallest possible PDU header).

Note:

This does NOT limit Rx PDUs, since the file data is written from the transport packet to the file.

Limits:

Since PDUs are wrapped in CCSDS packets, need to respect any CCSDS packet size limits on the system.

Definition at line 100 of file default_cf_interface_cfg.h.

10.5.2.10 CF_MAX_POLLING_DIR_PER_CHAN #define CF_MAX_POLLING_DIR_PER_CHAN (5)

Max number of polling directories per channel.

Description:

This affects the configuration table. There must be an entry (can be empty) for each of these polling directories per channel.

Limits:

Definition at line 79 of file default_cf_interface_cfg.h.

10.5.2.11 CF_MAX_SIMULTANEOUS_RX #define CF_MAX_SIMULTANEOUS_RX (5)

Max number of simultaneous file receives.

Description:

Each channel can support this number of file receive transactions at a time.

Limits:

Definition at line 130 of file default_cf_internal_cfg.h.

10.5.2.12 CF_NAK_MAX_SEGMENTS #define CF_NAK_MAX_SEGMENTS (58)

Max NAK segments supported in a NAK PDU.

Description:

When a NAK PDU is sent or received, this is the max number of segment requests supported. This number should match the ground CFDP engine configuration as well.

Limits:

Definition at line 67 of file default_cf_interface_cfg.h.

10.5.2.13 CF_NUM_CHANNELS #define CF_NUM_CHANNELS (2)

Number of channels.

Description:

The number of channels in the engine. Changing this value changes the configuration table for the application.

Limits:

Must be less <= 200. Obviously it will be smaller than that.

Definition at line 54 of file default_cf_interface_cfg.h.

10.5.2.14 CF_NUM_HISTORIES_PER_CHANNEL #define CF_NUM_HISTORIES_PER_CHANNEL (256)

Number of histories per channel.

Description:

Each channel can support this number of file receive transactions at a time.

Limits:

65536 is the current max.

Definition at line 154 of file default_cf_internal_cfg.h.

10.5.2.15 CF_NUM_TRANSACTIONS_PER_PLAYBACK #define CF_NUM_TRANSACTIONS_PER_PLAYBACK (5)

Number of transactions per playback directory.

Description:

Each playback/polling directory operation will be able to have this many active transfers at a time pending or active.

Limits:

Definition at line 166 of file default_cf_internal_cfg.h.

10.5.2.16 CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES #define CF_PDU_ENCAPSULATION_EXTRA_T←

RAILING_BYTES 0

Number of trailing bytes to add to CFDP PDU.

Description

Additional padding bytes to be appended to the tail of CFDP PDUs. This reserves extra space to the software bus encapsulation buffer for every CFDP PDU such that platform-specific trailer information may be added. This includes, but is not limited to a separate CRC or error control field in addition to the error control field(s) within the the nominal CFDP protocol.

These extra bytes are added at the software bus encapsulation layer, they are not part of the CFDP PDU itself. Set to 0 to disable this feature, such that the software bus buffer encapsulates only the CFDP PDU and no extra bytes are added.

Limits:

Maximum value is the difference between the maximum size of a CFDP PDU and the maximum size of an SB message.

Definition at line 138 of file default_cf_interface_cfg.h.

10.5.2.17 CF_PIPE_DEPTH #define CF_PIPE_DEPTH (32)
Application Pipe Depth.

Description:

Dictates the pipe depth of the cf command pipe.

Limits:

The minimum size of this parameter is 1 The maximum size dictated by cFE platform configuration parameter is OS_QUEUE_MAX_DEPTH

Definition at line 108 of file default_cf_internal_cfg.h.

10.5.2.18 CF_R2_CRC_CHUNK_SIZE #define CF_R2_CRC_CHUNK_SIZE (1024)
R2 CRC calc chunk size.

Description

R2 performs CRC calculation upon file completion in chunks. This is the size of the buffer. The larger the size the more stack will be used, but the faster it can go. The overall number of bytes calculated per wakeup is set in the configuration table.

Limits:

Definition at line 204 of file default_cf_internal_cfg.h.

10.5.2.19 CF_RCVMSG_TIMEOUT #define CF_RCVMSG_TIMEOUT (100)
Number of milliseconds to wait for a SB message.
Definition at line 209 of file default_cf_internal_cfg.h.

10.5.2.20 CF_STARTUP_SEM_MAX_RETRIES #define CF_STARTUP_SEM_MAX_RETRIES 25
Limits the number of retries to obtain the CF throttle sem.

Description

If the CF throttle sem is not available during CF startup, the initialization will retry after a short delay.

See also

[CF_STARTUP_SEM_TASK_DELAY](#)

Definition at line 220 of file default_cf_internal_cfg.h.

10.5.2.21 CF_STARTUP_SEM_TASK_DELAY #define CF_STARTUP_SEM_TASK_DELAY 100
Number of milliseconds to wait if CF throttle sem is not available.

Description

If the CF throttle sem is not available during CF startup, the initialization will delay for this period of time before trying again

See also

[CF_STARTUP_SEM_MAX_RETRIES](#)

Definition at line 231 of file default_cf_internal_cfg.h.

10.6 CFS CFDP Telemetry

Data Structures

- struct [CF_HkCmdCounters](#)
Housekeeping command counters.
- struct [CF_HkSent](#)
Housekeeping sent counters.
- struct [CF_HkRecv](#)
Housekeeping received counters.
- struct [CF_HkFault](#)
Housekeeping fault counters.
- struct [CF_HkCounters](#)
Housekeeping counters.
- struct [CF_HkChannel_Data](#)
Housekeeping channel data.
- struct [CF_HkPacket_Payload](#)
Housekeeping packet.
- struct [CF_EotPacket_Payload](#)
End of transaction packet.
- struct [CF_HkPacket](#)
Housekeeping packet.
- struct [CF_EotPacket](#)
End of transaction packet.

Typedefs

- typedef struct [CF_HkCmdCounters](#) [CF_HkCmdCounters_t](#)
Housekeeping command counters.
- typedef struct [CF_HkSent](#) [CF_HkSent_t](#)
Housekeeping sent counters.
- typedef struct [CF_HkRecv](#) [CF_HkRecv_t](#)
Housekeeping received counters.
- typedef struct [CF_HkFault](#) [CF_HkFault_t](#)
Housekeeping fault counters.
- typedef struct [CF_HkCounters](#) [CF_HkCounters_t](#)
Housekeeping counters.
- typedef struct [CF_HkChannel_Data](#) [CF_HkChannel_Data_t](#)
Housekeeping channel data.
- typedef struct [CF_HkPacket_Payload](#) [CF_HkPacket_Payload_t](#)
Housekeeping packet.
- typedef struct [CF_EotPacket_Payload](#) [CF_EotPacket_Payload_t](#)
End of transaction packet.
- typedef struct [CF_HkPacket](#) [CF_HkPacket_t](#)
Housekeeping packet.
- typedef struct [CF_EotPacket](#) [CF_EotPacket_t](#)
End of transaction packet.

10.6.1 Detailed Description**10.6.2 Typedef Documentation**

10.6.2.1 CF_EotPacket_Payload_t `typedef struct CF_EotPacket_Payload CF_EotPacket_Payload_t`
End of transaction packet.

10.6.2.2 CF_EotPacket_t `typedef struct CF_EotPacket CF_EotPacket_t`
End of transaction packet.

10.6.2.3 CF_HkChannel_Data_t `typedef struct CF_HkChannel_Data CF_HkChannel_Data_t`
Housekeeping channel data.

10.6.2.4 CF_HkCmdCounters_t `typedef struct CF_HkCmdCounters CF_HkCmdCounters_t`
Housekeeping command counters.

10.6.2.5 CF_HkCounters_t `typedef struct CF_HkCounters CF_HkCounters_t`
Housekeeping counters.

10.6.2.6 CF_HkFault_t `typedef struct CF_HkFault CF_HkFault_t`
Housekeeping fault counters.

10.6.2.7 CF_HkPacket_Payload_t `typedef struct CF_HkPacket_Payload CF_HkPacket_Payload_t`
Housekeeping packet.

10.6.2.8 CF_HkPacket_t `typedef struct CF_HkPacket CF_HkPacket_t`
Housekeeping packet.

10.6.2.9 CF_HkRecv_t `typedef struct CF_HkRecv CF_HkRecv_t`
Housekeeping received counters.

10.6.2.10 CF_HkSent_t `typedef struct CF_HkSent CF_HkSent_t`
Housekeeping sent counters.

10.7 CFS CFDP Command Structures

Data Structures

- union [CF_UnionArgs_Payload](#)
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.
- struct [CF_GetParam_Payload](#)
Get parameter command structure.
- struct [CF_SetParam_Payload](#)
Set parameter command structure.
- struct [CF_TxFile_Payload](#)
Transmit file command structure.
- struct [CF_WriteQueue_Payload](#)
Write Queue command structure.
- struct [CF_Transaction_Payload](#)
Transaction command structure.
- struct [CF_NoopCmd](#)
Noop command structure.
- struct [CF_EnableEngineCmd](#)
EnableEngine command structure.
- struct [CF_DisableEngineCmd](#)
DisableEngine command structure.
- struct [CF_ResetCountersCmd](#)
Reset command structure.
- struct [CF_FreezeCmd](#)
Freeze command structure.
- struct [CF_ThawCmd](#)
Thaw command structure.
- struct [CF_EnableDequeueCmd](#)
EnableDequeue command structure.
- struct [CF_DisableDequeueCmd](#)
DisableDequeue command structure.
- struct [CF_EnableDirPollingCmd](#)
EnableDirPolling command structure.
- struct [CF_DisableDirPollingCmd](#)
DisableDirPolling command structure.
- struct [CF_PurgeQueueCmd](#)
PurgeQueue command structure.
- struct [CF_GetParamCmd](#)
Get parameter command structure.
- struct [CF_SetParamCmd](#)
Set parameter command structure.
- struct [CF_TxFileCmd](#)
Transmit file command structure.
- struct [CF_WriteQueueCmd](#)
Write Queue command structure.
- struct [CF_PlaybackDirCmd](#)
Playback directory command structure.

- struct **CF_SuspendCmd**
Suspend command structure.
- struct **CF_ResumeCmd**
Resume command structure.
- struct **CF_CancelCmd**
Cancel command structure.
- struct **CF_AbandonCmd**
Abandon command structure.
- struct **CF_SendHkCmd**
Send Housekeeping Command.
- struct **CF_WakeupCmd**
Wake Up Command.

Typedefs

- typedef union **CF_UnionArgs_Payload** **CF_UnionArgs_Payload_t**
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.
- typedef struct **CF_GetParam_Payload** **CF_GetParam_Payload_t**
Get parameter command structure.
- typedef struct **CF_SetParam_Payload** **CF_SetParam_Payload_t**
Set parameter command structure.
- typedef struct **CF_TxFile_Payload** **CF_TxFile_Payload_t**
Transmit file command structure.
- typedef struct **CF_WriteQueue_Payload** **CF_WriteQueue_Payload_t**
Write Queue command structure.
- typedef struct **CF_Transaction_Payload** **CF_Transaction_Payload_t**
Transaction command structure.
- typedef struct **CF_NoopCmd** **CF_NoopCmd_t**
Noop command structure.
- typedef struct **CF_EnableEngineCmd** **CF_EnableEngineCmd_t**
EnableEngine command structure.
- typedef struct **CF_DisableEngineCmd** **CF_DisableEngineCmd_t**
DisableEngine command structure.
- typedef struct **CF_ResetCountersCmd** **CF_ResetCountersCmd_t**
Reset command structure.
- typedef struct **CF_FreezeCmd** **CF_FreezeCmd_t**
Freeze command structure.
- typedef struct **CF_ThawCmd** **CF_ThawCmd_t**
Thaw command structure.
- typedef struct **CF_EnableDequeueCmd** **CF_EnableDequeueCmd_t**
EnableDequeue command structure.
- typedef struct **CF_DisableDequeueCmd** **CF_DisableDequeueCmd_t**
DisableDequeue command structure.
- typedef struct **CF_EnableDirPollingCmd** **CF_EnableDirPollingCmd_t**
EnableDirPolling command structure.
- typedef struct **CF_DisableDirPollingCmd** **CF_DisableDirPollingCmd_t**
DisableDirPolling command structure.
- typedef struct **CF_PurgeQueueCmd** **CF_PurgeQueueCmd_t**

PurgeQueue command structure.

- **typedef struct CF_GetParamCmd CF_GetParamCmd_t**
Get parameter command structure.
- **typedef struct CF_SetParamCmd CF_SetParamCmd_t**
Set parameter command structure.
- **typedef struct CF_TxFileCmd CF_TxFileCmd_t**
Transmit file command structure.
- **typedef struct CF_WriteQueueCmd CF_WriteQueueCmd_t**
Write Queue command structure.
- **typedef struct CF_PlaybackDirCmd CF_PlaybackDirCmd_t**
Playback directory command structure.
- **typedef struct CF_SuspendCmd CF_SuspendCmd_t**
Suspend command structure.
- **typedef struct CF_ResumeCmd CF_ResumeCmd_t**
Resume command structure.
- **typedef struct CF_CancelCmd CF_CancelCmd_t**
Cancel command structure.
- **typedef struct CF_AbandonCmd CF_AbandonCmd_t**
Abandon command structure.
- **typedef struct CF_SendHkCmd CF_SendHkCmd_t**
Send Housekeeping Command.
- **typedef struct CF_WakeupCmd CF_WakeupCmd_t**
Wake Up Command.

Enumerations

- **enum CF_Reset_t {**
 CF_Reset_all = 0, CF_Reset_command = 1, CF_Reset_fault = 2, CF_Reset_up = 3,
 CF_Reset_down = 4 }
IDs for use for Reset cmd.
- **enum CF_Type_t { CF_Type_all = 0, CF_Type_up = 1, CF_Type_down = 2 }**
Type IDs for use for Write Queue cmd.
- **enum CF_Queue_t { CF_Queue_pend = 0, CF_Queue_active = 1, CF_Queue_history = 2, CF_Queue_all = 3 }**
Queue IDs for use for Write Queue cmd.
- **enum CF_GetSet_ValueID_t {**
 CF_GetSet_ValueID_ticks_per_second, CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup, CF_GetSet_ValueID_ack_timer_s,
 CF_GetSet_ValueID_nak_timer_s,
 CF_GetSet_ValueID_inactivity_timer_s, CF_GetSet_ValueID_outgoing_file_chunk_size, CF_GetSet_ValueID_ack_limit,
 CF_GetSet_ValueID_nak_limit,
 CF_GetSet_ValueID_local_eid, CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup, CF_GetSet_ValueID_MAX
}
Parameter IDs for use with Get/Set parameter messages.

10.7.1 Detailed Description

10.7.2 Typedef Documentation

10.7.2.1 CF_AbandonCmd_t `typedef struct CF_AbandonCmd CF_AbandonCmd_t`
Abandon command structure.

For command details see [CF_ABANDON_CC](#)

10.7.2.2 CF_CancelCmd_t `typedef struct CF_CancelCmd CF_CancelCmd_t`
Cancel command structure.

For command details see [CF_CANCEL_CC](#)

10.7.2.3 CF_DisableDequeueCmd_t `typedef struct CF_DisableDequeueCmd CF_DisableDequeueCmd_t`
DisableDequeue command structure.

For command details see [CF_DISABLE_DEQUEUE_CC](#)

10.7.2.4 CF_DisableDirPollingCmd_t `typedef struct CF_DisableDirPollingCmd CF_DisableDirPollingCmd_t`
DisableDirPolling command structure.

For command details see [CF_DISABLE_DIR_POLLING_CC](#)

10.7.2.5 CF_DisableEngineCmd_t `typedef struct CF_DisableEngineCmd CF_DisableEngineCmd_t`
DisableEngine command structure.

For command details see [CF_DISABLE_ENGINE_CC](#)

10.7.2.6 CF_EnableDequeueCmd_t `typedef struct CF_EnableDequeueCmd CF_EnableDequeueCmd_t`
EnableDequeue command structure.

For command details see [CF_ENABLE_DEQUEUE_CC](#)

10.7.2.7 CF_EnableDirPollingCmd_t `typedef struct CF_EnableDirPollingCmd CF_EnableDirPollingCmd_t`
EnableDirPolling command structure.

For command details see [CF_ENABLE_DIR_POLLING_CC](#)

10.7.2.8 CF_EnableEngineCmd_t `typedef struct CF_EnableEngineCmd CF_EnableEngineCmd_t`
EnableEngine command structure.

For command details see [CF_ENABLE_ENGINE_CC](#)

10.7.2.9 CF_FreezeCmd_t `typedef struct CF_FreezeCmd CF_FreezeCmd_t`
Freeze command structure.

For command details see [CF_FREEZE_CC](#)

10.7.2.10 CF_GetParam_Payload_t `typedef struct CF_GetParam_Payload CF_GetParam_Payload_t`
Get parameter command structure.

For command details see [CF_GET_PARAM_CC](#)

10.7.2.11 CF_GetParamCmd_t `typedef struct CF_GetParamCmd CF_GetParamCmd_t`
Get parameter command structure.

For command details see [CF_GET_PARAM_CC](#)

10.7.2.12 CF_NoopCmd_t `typedef struct CF_NoopCmd CF_NoopCmd_t`
Noop command structure.

For command details see [CF_NOOP_CC](#)

10.7.2.13 CF_PlaybackDirCmd_t `typedef struct CF_PlaybackDirCmd CF_PlaybackDirCmd_t`
Playback directory command structure.

For command details see [CF_PLAYBACK_DIR_CC](#)

10.7.2.14 CF_PurgeQueueCmd_t `typedef struct CF_PurgeQueueCmd CF_PurgeQueueCmd_t`
PurgeQueue command structure.

For command details see [CF_PURGE_QUEUE_CC](#)

10.7.2.15 CF_ResetCountersCmd_t `typedef struct CF_ResetCountersCmd CF_ResetCountersCmd_t`
Reset command structure.

For command details see [CF_RESET_CC](#)

10.7.2.16 CF_ResumeCmd_t `typedef struct CF_ResumeCmd CF_ResumeCmd_t`
Resume command structure.

For command details see [CF_RESUME_CC](#)

10.7.2.17 CF_SendHkCmd_t `typedef struct CF_SendHkCmd CF_SendHkCmd_t`
Send Housekeeping Command.

Internal notification from SCH with no payload

10.7.2.18 CF_SetParam_Payload_t `typedef struct CF_SetParam_Payload CF_SetParam_Payload_t`
Set parameter command structure.

For command details see [CF_SET_PARAM_CC](#)

10.7.2.19 CF_SetParamCmd_t `typedef struct CF_SetParamCmd CF_SetParamCmd_t`
Set parameter command structure.

For command details see [CF_SET_PARAM_CC](#)

10.7.2.20 CF_SuspendCmd_t `typedef struct CF_SuspendCmd CF_SuspendCmd_t`
Suspend command structure.

For command details see [CF_SUSPEND_CC](#)

10.7.2.21 CF_ThawCmd_t `typedef struct CF_ThawCmd CF_ThawCmd_t`
Thaw command structure.

For command details see [CF_THAW_CC](#)

10.7.2.22 CF_Transaction_Payload_t `typedef struct CF_Transaction_Payload CF_Transaction_Payload_t`
Transaction command structure.

For command details see [CF_SUSPEND_CC, CF_RESUME_CC, CF_CANCEL_CC, CF_ABANDON_CC](#)

10.7.2.23 CF_TxFile_Payload_t `typedef struct CF_TxFile_Payload CF_TxFile_Payload_t`
Transmit file command structure.

For command details see [CF_TX_FILE_CC](#)

10.7.2.24 CF_TxFileCmd_t `typedef struct CF_TxFileCmd CF_TxFileCmd_t`
Transmit file command structure.

For command details see [CF_TX_FILE_CC](#)

10.7.2.25 CF_UnionArgs_Payload_t `typedef union CF_UnionArgs_Payload CF_UnionArgs_Payload_t`
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

10.7.2.26 CF_WakeupCmd_t `typedef struct CF_WakeupCmd CF_WakeupCmd_t`
Wake Up Command.

Internal notification from SCH with no payload

10.7.2.27 CF_WriteQueue_Payload_t `typedef struct CF_WriteQueue_Payload CF_WriteQueue_Payload_t`
 Write Queue command structure.

For command details see [CF_WRITE_QUEUE_CC](#)

10.7.2.28 CF_WriteQueueCmd_t `typedef struct CF_WriteQueueCmd CF_WriteQueueCmd_t`
 Write Queue command structure.

For command details see [CF_WRITE_QUEUE_CC](#)

10.7.3 Enumeration Type Documentation

10.7.3.1 CF_GetSet_ValueID_t `enum CF_GetSet_ValueID_t`

Parameter IDs for use with Get/Set parameter messages.

Specifically these are used for the "key" field within CF_SetParamCmd_t and CF_SetParamCmd_t message structures.

Enumerator

CF_GetSet_ValueID_ticks_per_second	Ticks per second key.
CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup	Receive CRC calculated bytes per wake-up key.
CF_GetSet_ValueID_ack_timer_s	ACK timer in seconds key.
CF_GetSet_ValueID_nak_timer_s	NAK timer in seconds key.
CF_GetSet_ValueID_inactivity_timer_s	Inactivity timer in seconds key.
CF_GetSet_ValueID_outgoing_file_chunk_size	Outgoing file chunk size key.
CF_GetSet_ValueID_ack_limit	ACK retry limit key.
CF_GetSet_ValueID_nak_limit	NAK retry limit key.
CF_GetSet_ValueID_local_eid	Local entity id key.
CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup	Max outgoing messages per wake-up key.
CF_GetSet_ValueID_MAX	Key limit used for validity check.

Definition at line 197 of file default_cf_msgdefs.h.

10.7.3.2 CF_Queue_t `enum CF_Queue_t`

Queue IDs for use for Write Queue cmd.

Enumerator

CF_Queue_pend	Queue pending.
CF_Queue_active	Queue active.
CF_Queue_history	Queue history.
CF_Queue_all	Queue all.

Definition at line 183 of file default_cf_msgdefs.h.

10.7.3.3 CF_Reset_t `enum CF_Reset_t`

IDs for use for Reset cmd.

Enumerator

CF_Reset_all	Reset all.
CF_Reset_command	Reset command.
CF_Reset_fault	Reset fault.
CF_Reset_up	Reset up.
CF_Reset_down	Reset down.

Definition at line 161 of file default_cf_msgdefs.h.

10.7.3.4 CF_Type_t enum CF_Type_t

Type IDs for use for Write Queue cmd.

Enumerator

CF_Type_all	Type all.
CF_Type_up	Type up.
CF_Type_down	Type down.

Definition at line 173 of file default_cf_msgdefs.h.

10.8 CFS CFDP Command Message IDs

Macros

- `#define CF_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CMD_TOPICID)`
Message ID for commands.
- `#define CF_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_SEND_HK_TOPICID)`
Message ID to request housekeeping telemetry.
- `#define CF_WAKE_UP_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_WAKE_UP_TOPICID)`
Message ID for waking up the processing cycle.

10.8.1 Detailed Description

10.8.2 Macro Definition Documentation

10.8.2.1 CF_CMD_MID `#define CF_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CMD_TOPICID)`
Message ID for commands.

Definition at line 36 of file default_cf_msgids.h.

10.8.2.2 CF_SEND_HK_MID `#define CF_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_SEND_HK_TOPICID)`
Message ID to request housekeeping telemetry.

Definition at line 39 of file default_cf_msgids.h.

10.8.2.3 CF_WAKE_UP_MID `#define CF_WAKE_UP_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_WAKE_UP_TOPICID)`
Message ID for waking up the processing cycle.

Definition at line 42 of file default_cf_msgids.h.

10.9 CFS CFDP Telemetry Message IDs

Macros

- `#define CF_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_HK_TLM_TOPICID)`
Message ID for housekeeping telemetry.
- `#define CF_EOT_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_EOT_TLM_TOPICID)`
Message ID for end of transaction telemetry.

10.9.1 Detailed Description

10.9.2 Macro Definition Documentation

10.9.2.1 CF_EOT_TLM_MID `#define CF_EOT_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_EOT_TLM_TOPICID)`

Message ID for end of transaction telemetry.

Definition at line 55 of file default_cf_msgids.h.

10.9.2.2 CF_HK_TLM_MID `#define CF_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_HK_TLM_TOPICID)`

Message ID for housekeeping telemetry.

Definition at line 52 of file default_cf_msgids.h.

10.10 CFS CFDP Data Interface Message IDs

Macros

- #define CF_CH0_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_TX_TOPICID)
- #define CF_CH1_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_TX_TOPICID)
- #define CF_CH0_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_RX_TOPICID)
- #define CF_CH1_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_RX_TOPICID)

10.10.1 Detailed Description

10.10.2 Macro Definition Documentation

10.10.2.1 CF_CH0_RX_MID #define CF_CH0_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_RX_TOPICID)
Definition at line 66 of file default_cf_msgids.h.

10.10.2.2 CF_CH0_TX_MID #define CF_CH0_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_TX_TOPICID)
Definition at line 64 of file default_cf_msgids.h.

10.10.2.3 CF_CH1_RX_MID #define CF_CH1_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_RX_TOPICID)
Definition at line 67 of file default_cf_msgids.h.

10.10.2.4 CF_CH1_TX_MID #define CF_CH1_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_TX_TOPICID)
Definition at line 65 of file default_cf_msgids.h.

10.11 cFE Return Code Defines

Macros

- #define CFE_SUCCESS ((CFE_Status_t)0)
Successful execution.
- #define CFE_STATUS_NO_COUNTER_INCREMENT ((CFE_Status_t)0x48000001)
No Counter Increment.
- #define CFE_STATUS_WRONG_MSG_LENGTH ((CFE_Status_t)0xc8000002)
Wrong Message Length.
- #define CFE_STATUS_UNKNOWN_MSG_ID ((CFE_Status_t)0xc8000003)
Unknown Message ID.
- #define CFE_STATUS_BAD_COMMAND_CODE ((CFE_Status_t)0xc8000004)
Bad Command Code.
- #define CFE_STATUS_EXTERNAL_RESOURCE_FAIL ((CFE_Status_t)0xc8000005)
External failure.
- #define CFE_STATUS_REQUEST_ALREADY_PENDING ((int32)0xc8000006)
Request already pending.
- #define CFE_STATUS_VALIDATION_FAILURE ((int32)0xc8000007)
Request or input value failed basic structural validation.
- #define CFE_STATUS_RANGE_ERROR ((int32)0xc8000008)
Request or input value is out of range.
- #define CFE_STATUS_INCORRECT_STATE ((int32)0xc8000009)
Cannot process request at this time.
- #define CFE_STATUS_NOT_IMPLEMENTED ((CFE_Status_t)0xc800ffff)
Not Implemented.
- #define CFE_EVS_UNKNOWN_FILTER ((CFE_Status_t)0xc2000001)
Unknown Filter.
- #define CFE_EVS_APP_NOT_REGISTERED ((CFE_Status_t)0xc2000002)
Application Not Registered.
- #define CFE_EVS_APP_ILLEGAL_APP_ID ((CFE_Status_t)0xc2000003)
Illegal Application ID.
- #define CFE_EVS_APP_FILTER_OVERLOAD ((CFE_Status_t)0xc2000004)
Application Filter Overload.
- #define CFE_EVS_RESET_AREA_POINTER ((CFE_Status_t)0xc2000005)
Reset Area Pointer Failure.
- #define CFE_EVS_EVT_NOT_REGISTERED ((CFE_Status_t)0xc2000006)
Event Not Registered.
- #define CFE_EVS_FILE_WRITE_ERROR ((CFE_Status_t)0xc2000007)
File Write Error.
- #define CFE_EVS_INVALID_PARAMETER ((CFE_Status_t)0xc2000008)
Invalid Pointer.
- #define CFE_EVS_APP_SQUELCHED ((CFE_Status_t)0xc2000009)
Event squelched.
- #define CFE_EVS_NOT_IMPLEMENTED ((CFE_Status_t)0xc200ffff)
Not Implemented.
- #define CFE_ES_ERR_RESOURCEID_NOT_VALID ((CFE_Status_t)0xc4000001)
Resource ID is not valid.

- #define CFE_ES_ERR_NAME_NOT_FOUND ((CFE_Status_t)0xc4000002)
Resource Name Error.
- #define CFE_ES_ERR_APP_CREATE ((CFE_Status_t)0xc4000004)
Application Create Error.
- #define CFE_ES_ERR_CHILD_TASK_CREATE ((CFE_Status_t)0xc4000005)
Child Task Create Error.
- #define CFE_ES_ERR_SYS_LOG_FULL ((CFE_Status_t)0xc4000006)
System Log Full.
- #define CFE_ES_ERR_MEM_BLOCK_SIZE ((CFE_Status_t)0xc4000008)
Memory Block Size Error.
- #define CFE_ES_ERR_LOAD_LIB ((CFE_Status_t)0xc4000009)
Load Library Error.
- #define CFE_ES_BAD_ARGUMENT ((CFE_Status_t)0xc400000a)
Bad Argument.
- #define CFE_ES_ERR_CHILD_TASK_REGISTER ((CFE_Status_t)0xc400000b)
Child Task Register Error.
- #define CFE_ES_CDS_ALREADY_EXISTS ((CFE_Status_t)0x4400000d)
CDS Already Exists.
- #define CFE_ES_CDS_INSUFFICIENT_MEMORY ((CFE_Status_t)0xc400000e)
CDS Insufficient Memory.
- #define CFE_ES_CDS_INVALID_NAME ((CFE_Status_t)0xc400000f)
CDS Invalid Name.
- #define CFE_ES_CDS_INVALID_SIZE ((CFE_Status_t)0xc4000010)
CDS Invalid Size.
- #define CFE_ES_CDS_INVALID ((CFE_Status_t)0xc4000012)
CDS Invalid.
- #define CFE_ES_CDS_ACCESS_ERROR ((CFE_Status_t)0xc4000013)
CDS Access Error.
- #define CFE_ES_FILE_IO_ERR ((CFE_Status_t)0xc4000014)
File IO Error.
- #define CFE_ES_RST_ACCESS_ERR ((CFE_Status_t)0xc4000015)
Reset Area Access Error.
- #define CFE_ES_ERR_APP_REGISTER ((CFE_Status_t)0xc4000017)
Application Register Error.
- #define CFE_ES_ERR_CHILD_TASK_DELETE ((CFE_Status_t)0xc4000018)
Child Task Delete Error.
- #define CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK ((CFE_Status_t)0xc4000019)
Child Task Delete Passed Main Task.
- #define CFE_ES_CDS_BLOCK_CRC_ERR ((CFE_Status_t)0xc400001A)
CDS Block CRC Error.
- #define CFE_ES_MUT_SEM_DELETE_ERR ((CFE_Status_t)0xc400001B)
Mutex Semaphore Delete Error.
- #define CFE_ES_BIN_SEM_DELETE_ERR ((CFE_Status_t)0xc400001C)
Binary Semaphore Delete Error.
- #define CFE_ES_COUNT_SEM_DELETE_ERR ((CFE_Status_t)0xc400001D)
Counting Semaphore Delete Error.
- #define CFE_ES_QUEUE_DELETE_ERR ((CFE_Status_t)0xc400001E)

- #define CFE_ES_FILE_CLOSE_ERR ((CFE_Status_t)0xc400001F)
File Close Error.
- #define CFE_ES_CDS_WRONG_TYPE_ERR ((CFE_Status_t)0xc4000020)
CDS Wrong Type Error.
- #define CFE_ES_CDS_OWNER_ACTIVE_ERR ((CFE_Status_t)0xc4000022)
CDS Owner Active Error.
- #define CFE_ES_APP_CLEANUP_ERR ((CFE_Status_t)0xc4000023)
Application Cleanup Error.
- #define CFE_ES_TIMER_DELETE_ERR ((CFE_Status_t)0xc4000024)
Timer Delete Error.
- #define CFE_ES_BUFFER_NOT_IN_POOL ((CFE_Status_t)0xc4000025)
Buffer Not In Pool.
- #define CFE_ES_TASK_DELETE_ERR ((CFE_Status_t)0xc4000026)
Task Delete Error.
- #define CFE_ES_OPERATION_TIMED_OUT ((CFE_Status_t)0xc4000027)
Operation Timed Out.
- #define CFE_ES_LIB_ALREADY_LOADED ((CFE_Status_t)0x44000028)
Library Already Loaded.
- #define CFE_ES_ERR_SYS_LOG_TRUNCATED ((CFE_Status_t)0x44000029)
System Log Message Truncated.
- #define CFE_ES_NO_RESOURCE_IDS_AVAILABLE ((CFE_Status_t)0xc400002B)
Resource ID is not available.
- #define CFE_ES_POOL_BLOCK_INVALID ((CFE_Status_t)0xc400002C)
Invalid pool block.
- #define CFE_ES_ERR_DUPLICATE_NAME ((CFE_Status_t)0xc400002E)
Duplicate Name Error.
- #define CFE_ES_NOT_IMPLEMENTED ((CFE_Status_t)0xc400ffff)
Not Implemented.
- #define CFE_FS_BAD_ARGUMENT ((CFE_Status_t)0xc6000001)
Bad Argument.
- #define CFE_FS_INVALID_PATH ((CFE_Status_t)0xc6000002)
Invalid Path.
- #define CFE_FS_FNAME_TOO_LONG ((CFE_Status_t)0xc6000003)
Filename Too Long.
- #define CFE_FS_NOT_IMPLEMENTED ((CFE_Status_t)0xc600ffff)
Not Implemented.
- #define CFE_SB_TIME_OUT ((CFE_Status_t)0xca000001)
Time Out.
- #define CFE_SB_NO_MESSAGE ((CFE_Status_t)0xca000002)
No Message.
- #define CFE_SB_BAD_ARGUMENT ((CFE_Status_t)0xca000003)
Bad Argument.
- #define CFE_SB_MAX_PIPES_MET ((CFE_Status_t)0xca000004)
Max Pipes Met.
- #define CFE_SB_PIPE_CR_ERR ((CFE_Status_t)0xca000005)
Pipe Create Error.

- #define CFE_SB_PIPE_RD_ERR ((CFE_Status_t)0xca000006)
Pipe Read Error.
- #define CFE_SB_MSG_TOO_BIG ((CFE_Status_t)0xca000007)
Message Too Big.
- #define CFE_SB_BUF_ALOC_ERR ((CFE_Status_t)0xca000008)
Buffer Allocation Error.
- #define CFE_SB_MAX_MSGS_MET ((CFE_Status_t)0xca000009)
Max Messages Met.
- #define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca00000a)
Max Destinations Met.
- #define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca00000c)
Internal Error.
- #define CFE_SB_WRONG_MSG_TYPE ((CFE_Status_t)0xca00000d)
Wrong Message Type.
- #define CFE_SB_BUFFER_INVALID ((CFE_Status_t)0xca00000e)
Buffer Invalid.
- #define CFE_SB_NOT_IMPLEMENTED ((CFE_Status_t)0xca00ffff)
Not Implemented.
- #define CFE_TBL_ERR_INVALID_HANDLE ((CFE_Status_t)0xcc000001)
Invalid Handle.
- #define CFE_TBL_ERR_INVALID_NAME ((CFE_Status_t)0xcc000002)
Invalid Name.
- #define CFE_TBL_ERR_INVALID_SIZE ((CFE_Status_t)0xcc000003)
Invalid Size.
- #define CFE_TBL_INFO_UPDATE_PENDING ((CFE_Status_t)0x4c000004)
Update Pending.
- #define CFE_TBL_ERR_NEVER_LOADED ((CFE_Status_t)0xcc000005)
Never Loaded.
- #define CFE_TBL_ERR_REGISTRY_FULL ((CFE_Status_t)0xcc000006)
Registry Full.
- #define CFE_TBL_WARN_DUPLICATE ((CFE_Status_t)0x4c000007)
Duplicate Warning.
- #define CFE_TBL_ERR_NO_ACCESS ((CFE_Status_t)0xcc000008)
No Access.
- #define CFE_TBL_ERR_UNREGISTERED ((CFE_Status_t)0xcc000009)
Unregistered.
- #define CFE_TBL_ERR_HANDLES_FULL ((CFE_Status_t)0xcc00000B)
Handles Full.
- #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((CFE_Status_t)0xcc00000C)
Duplicate Table With Different Size.
- #define CFE_TBL_ERR_DUPLICATE_NOT OWNED ((CFE_Status_t)0xcc00000D)
Duplicate Table And Not Owned.
- #define CFE_TBL_INFO_UPDATED ((CFE_Status_t)0x4c00000E)
Updated.
- #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((CFE_Status_t)0xcc00000F)
No Buffer Available.
- #define CFE_TBL_ERR_DUMP_ONLY ((CFE_Status_t)0xcc000010)

- Dump Only Error.*
- #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((CFE_Status_t)0xcc000011)
Illegal Source Type.
 - #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((CFE_Status_t)0xcc000012)
Load In Progress.
 - #define CFE_TBL_ERR_FILE_TOO_LARGE ((CFE_Status_t)0xcc000014)
File Too Large.
 - #define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)
Short File Warning.
 - #define CFE_TBL_ERR_BAD_CONTENT_ID ((CFE_Status_t)0xcc000016)
Bad Content ID.
 - #define CFE_TBL_INFO_NO_UPDATE_PENDING ((CFE_Status_t)0x4c000017)
No Update Pending.
 - #define CFE_TBL_INFO_TABLE_LOCKED ((CFE_Status_t)0x4c000018)
Table Locked.
 - #define CFE_TBL_INFO_VALIDATION_PENDING ((CFE_Status_t)0x4c000019)
 - #define CFE_TBL_INFO_NO_VALIDATION_PENDING ((CFE_Status_t)0x4c00001A)
 - #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((CFE_Status_t)0xcc00001B)
Bad Subtype ID.
 - #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((CFE_Status_t)0xcc00001C)
File Size Inconsistent.
 - #define CFE_TBL_ERR_NO_STD_HEADER ((CFE_Status_t)0xcc00001D)
No Standard Header.
 - #define CFE_TBL_ERR_NO_TBL_HEADER ((CFE_Status_t)0xcc00001E)
No Table Header.
 - #define CFE_TBL_ERR_FILENAME_TOO_LONG ((CFE_Status_t)0xcc00001F)
Filename Too Long.
 - #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc000020)
File For Wrong Table.
 - #define CFE_TBL_ERR_LOAD_INCOMPLETE ((CFE_Status_t)0xcc000021)
Load Incomplete.
 - #define CFE_TBL_WARN_PARTIAL_LOAD ((CFE_Status_t)0x4c000022)
Partial Load Warning.
 - #define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)
Partial Load Error.
 - #define CFE_TBL_INFO_DUMP_PENDING ((CFE_Status_t)0x4c000024)
Dump Pending.
 - #define CFE_TBL_ERR_INVALID_OPTIONS ((CFE_Status_t)0xcc000025)
Invalid Options.
 - #define CFE_TBL_WARN_NOT_CRITICAL ((CFE_Status_t)0x4c000026)
Not Critical Warning.
 - #define CFE_TBL_INFO_RECOVERED_TBL ((CFE_Status_t)0x4c000027)
Recovered Table.
 - #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((CFE_Status_t)0xcc000028)
Bad Spacecraft ID.
 - #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((CFE_Status_t)0xcc000029)
Bad Processor ID.

- #define CFE_TBL_MESSAGE_ERROR ((CFE_Status_t)0xcc00002a)
Message Error.
- #define CFE_TBL_ERR_SHORT_FILE ((CFE_Status_t)0xcc00002b)
- #define CFE_TBL_ERR_ACCESS ((CFE_Status_t)0xcc00002c)
- #define CFE_TBL_BAD_ARGUMENT ((CFE_Status_t)0xcc00002d)
Bad Argument.
- #define CFE_TBL_NOT_IMPLEMENTED ((CFE_Status_t)0xcc00ffff)
Not Implemented.
- #define CFE_TIME_NOT_IMPLEMENTED ((CFE_Status_t)0xce00ffff)
Not Implemented.
- #define CFE_TIME_INTERNAL_ONLY ((CFE_Status_t)0xce000001)
Internal Only.
- #define CFE_TIME_OUT_OF_RANGE ((CFE_Status_t)0xce000002)
Out Of Range.
- #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((CFE_Status_t)0xce000003)
Too Many Sync Callbacks.
- #define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000004)
Callback Not Registered.
- #define CFE_TIME_BAD_ARGUMENT ((CFE_Status_t)0xce000005)
Bad Argument.

10.11.1 Detailed Description

10.11.2 Macro Definition Documentation

10.11.2.1 CFE_ES_APP_CLEANUP_ERR #define CFE_ES_APP_CLEANUP_ERR ((CFE_Status_t)0xc4000023)
Application Cleanup Error.

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 588 of file cfe_error.h.

10.11.2.2 CFE_ES_BAD_ARGUMENT #define CFE_ES_BAD_ARGUMENT ((CFE_Status_t)0xc40000a)
Bad Argument.

Bad parameter passed into an ES API.

Definition at line 399 of file cfe_error.h.

10.11.2.3 CFE_ES_BIN_SEM_DELETE_ERR #define CFE_ES_BIN_SEM_DELETE_ERR ((CFE_Status_t)0xc400001C)
Binary Semaphore Delete Error.

Occurs when trying to delete a Binary Semaphore that belongs to a task that ES is cleaning up.

Definition at line 527 of file cfe_error.h.

10.11.2.4 CFE_ES_BUFFER_NOT_IN_POOL #define CFE_ES_BUFFER_NOT_IN_POOL ((CFE_Status_t)0xc4000025)
Buffer Not In Pool.

The specified address is not in the memory pool.

Definition at line 605 of file cfe_error.h.

10.11.2.5 CFE_ES_CDS_ACCESS_ERROR #define CFE_ES_CDS_ACCESS_ERROR (([CFE_Status_t](#))0xc4000013)
CDS Access Error.

The CDS was inaccessible

Definition at line 458 of file cfe_error.h.

10.11.2.6 CFE_ES_CDS_ALREADY_EXISTS #define CFE_ES_CDS_ALREADY_EXISTS (([CFE_Status_t](#))0x4400000d)
CDS Already Exists.

The Application is receiving the pointer to a CDS that was already present.

Definition at line 415 of file cfe_error.h.

10.11.2.7 CFE_ES_CDS_BLOCK_CRC_ERR #define CFE_ES_CDS_BLOCK_CRC_ERR (([CFE_Status_t](#))0xc400001A)
CDS Block CRC Error.

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 509 of file cfe_error.h.

10.11.2.8 CFE_ES_CDS_INSUFFICIENT_MEMORY #define CFE_ES_CDS_INSUFFICIENT_MEMORY (([CFE_Status_t](#))0xc400000e)
CDS Insufficient Memory.

The Application is requesting a CDS Block that is larger than the remaining CDS memory.

Definition at line 424 of file cfe_error.h.

10.11.2.9 CFE_ES_CDS_INVALID #define CFE_ES_CDS_INVALID (([CFE_Status_t](#))0xc4000012)
CDS Invalid.

The CDS contents are invalid.

Definition at line 450 of file cfe_error.h.

10.11.2.10 CFE_ES_CDS_INVALID_NAME #define CFE_ES_CDS_INVALID_NAME (([CFE_Status_t](#))0xc400000f)
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_MAX_NAME_LENGTH](#)) or was an empty string.

Definition at line 433 of file cfe_error.h.

10.11.2.11 CFE_ES_CDS_INVALID_SIZE #define CFE_ES_CDS_INVALID_SIZE (([CFE_Status_t](#))0xc4000010)
CDS Invalid Size.

The Application is requesting a CDS Block or Pool with a size beyond the applicable limits, either too large or too small/zero.

Definition at line 442 of file cfe_error.h.

10.11.2.12 CFE_ES_CDS_OWNER_ACTIVE_ERR #define CFE_ES_CDS_OWNER_ACTIVE_ERR (([CFE_Status_t](#))0xc4000022)
CDS Owner Active Error.

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 575 of file cfe_error.h.

10.11.2.13 CFE_ES_CDS_WRONG_TYPE_ERR #define CFE_ES_CDS_WRONG_TYPE_ERR (([CFE_Status_t](#))0xc4000020)
CDS Wrong Type Error.

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 564 of file [cfe_error.h](#).

10.11.2.14 CFE_ES_COUNT_SEM_DELETE_ERR #define CFE_ES_COUNT_SEM_DELETE_ERR (([CFE_Status_t](#))0xc400001D)
Counting Semaphore Delete Error.

Occurs when trying to delete a Counting Semaphore that belongs to a task that ES is cleaning up.

Definition at line 536 of file [cfe_error.h](#).

10.11.2.15 CFE_ES_ERR_APP_CREATE #define CFE_ES_ERR_APP_CREATE (([CFE_Status_t](#))0xc4000004)
Application Create Error.

There was an error loading or creating the App.

Definition at line 358 of file [cfe_error.h](#).

10.11.2.16 CFE_ES_ERR_APP_REGISTER #define CFE_ES_ERR_APP_REGISTER (([CFE_Status_t](#))0xc4000017)
Application Register Error.

Occurs when a task cannot be registered in ES global tables

Definition at line 482 of file [cfe_error.h](#).

10.11.2.17 CFE_ES_ERR_CHILD_TASK_CREATE #define CFE_ES_ERR_CHILD_TASK_CREATE (([CFE_Status_t](#))0xc4000005)
Child Task Create Error.

There was an error creating a child task.

Definition at line 366 of file [cfe_error.h](#).

10.11.2.18 CFE_ES_ERR_CHILD_TASK_DELETE #define CFE_ES_ERR_CHILD_TASK_DELETE (([CFE_Status_t](#))0xc4000018)
Child Task Delete Error.

There was an error deleting a child task.

Definition at line 490 of file [cfe_error.h](#).

10.11.2.19 CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK #define CFE_ES_ERR_CHILD_TASK_DELETE_M←
AIN_TASK (([CFE_Status_t](#))0xc4000019)

Child Task Delete Passed Main Task.

There was an attempt to delete a cFE App Main Task with the [CFE_ES_DeleteChildTask](#) API.

Definition at line 499 of file [cfe_error.h](#).

10.11.2.20 CFE_ES_ERR_CHILD_TASK_REGISTER #define CFE_ES_ERR_CHILD_TASK_REGISTER (([CFE_Status_t](#))0xc400000b)
Child Task Register Error.

Errors occurred when trying to register a child task.

Definition at line 407 of file [cfe_error.h](#).

10.11.2.21 CFE_ES_ERR_DUPLICATE_NAME #define CFE_ES_ERR_DUPLICATE_NAME (([CFE_Status_t](#))0xc400002E)
Duplicate Name Error.

Resource creation failed due to the name already existing in the system.

Definition at line 668 of file cfe_error.h.

10.11.2.22 CFE_ES_ERR_LOAD_LIB #define CFE_ES_ERR_LOAD_LIB (([CFE_Status_t](#))0xc4000009)

Load Library Error.

Could not load the shared library.

Definition at line 391 of file cfe_error.h.

10.11.2.23 CFE_ES_ERR_MEM_BLOCK_SIZE #define CFE_ES_ERR_MEM_BLOCK_SIZE (([CFE_Status_t](#))0xc4000008)
Memory Block Size Error.

The block size requested is invalid.

Definition at line 383 of file cfe_error.h.

10.11.2.24 CFE_ES_ERR_NAME_NOT_FOUND #define CFE_ES_ERR_NAME_NOT_FOUND (([CFE_Status_t](#))0xc4000002)

Resource Name Error.

There is no match in the system for the given name.

Definition at line 350 of file cfe_error.h.

10.11.2.25 CFE_ES_ERR_RESOURCEID_NOT_VALID #define CFE_ES_ERR_RESOURCEID_NOT_VALID (([CFE_Status_t](#))0xc4000000)

Resource ID is not valid.

This error indicates that the passed in resource identifier (App ID, Lib ID, Counter ID, etc) did not validate.

Definition at line 342 of file cfe_error.h.

10.11.2.26 CFE_ES_ERR_SYS_LOG_FULL #define CFE_ES_ERR_SYS_LOG_FULL (([CFE_Status_t](#))0xc4000006)

System Log Full.

The cFE system Log is full. This error means the message was not logged at all

Definition at line 375 of file cfe_error.h.

10.11.2.27 CFE_ES_ERR_SYS_LOG_TRUNCATED #define CFE_ES_ERR_SYS_LOG_TRUNCATED (([CFE_Status_t](#))0x44000029)

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 640 of file cfe_error.h.

10.11.2.28 CFE_ES_FILE_CLOSE_ERR #define CFE_ES_FILE_CLOSE_ERR (([CFE_Status_t](#))0xc400001F)

File Close Error.

Occurs when trying to close a file that belongs to a task that ES is cleaning up.

Definition at line 554 of file cfe_error.h.

10.11.2.29 CFE_ES_FILE_IO_ERR #define CFE_ES_FILE_IO_ERR (([CFE_Status_t](#))0xc4000014)

File IO Error.

Occurs when a file operation fails

Definition at line 466 of file cfe_error.h.

10.11.2.30 CFE_ES_LIB_ALREADY_LOADED #define CFE_ES_LIB_ALREADY_LOADED ((CFE_Status_t)0x44000028)
Library Already Loaded.

Occurs if CFE_ES_LoadLibrary detects that the requested library name is already loaded.

Definition at line 631 of file cfe_error.h.

10.11.2.31 CFE_ES_MUT_SEM_DELETE_ERR #define CFE_ES_MUT_SEM_DELETE_ERR ((CFE_Status_t)0xc400001B)
Mutex Semaphore Delete Error.

Occurs when trying to delete a Mutex that belongs to a task that ES is cleaning up.

Definition at line 518 of file cfe_error.h.

10.11.2.32 CFE_ES_NO_RESOURCE_IDS_AVAILABLE #define CFE_ES_NO_RESOURCE_IDS_AVAILABLE ((CFE_Status_t)0xc400001C)
Resource ID is not available.

This error indicates that the maximum resource identifiers (App ID, Lib ID, Counter ID, etc) has already been reached and a new ID cannot be allocated.

Definition at line 650 of file cfe_error.h.

10.11.2.33 CFE_ES_NOT_IMPLEMENTED #define CFE_ES_NOT_IMPLEMENTED ((CFE_Status_t)0xc400ffff)
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.

Definition at line 679 of file cfe_error.h.

10.11.2.34 CFE_ES_OPERATION_TIMED_OUT #define CFE_ES_OPERATION_TIMED_OUT ((CFE_Status_t)0xc4000027)
Operation Timed Out.

Occurs if the timeout for a given operation was exceeded

Definition at line 622 of file cfe_error.h.

10.11.2.35 CFE_ES_POOL_BLOCK_INVALID #define CFE_ES_POOL_BLOCK_INVALID ((CFE_Status_t)0xc400002C)
Invalid pool block.

Software attempted to "put" a block back into a pool which does not appear to belong to that pool. This may mean the pool has become unusable due to memory corruption.

Definition at line 660 of file cfe_error.h.

10.11.2.36 CFE_ES_QUEUE_DELETE_ERR #define CFE_ES_QUEUE_DELETE_ERR ((CFE_Status_t)0xc400001E)
Queue Delete Error.

Occurs when trying to delete a Queue that belongs to a task that ES is cleaning up.

Definition at line 545 of file cfe_error.h.

10.11.2.37 CFE_ES_RST_ACCESS_ERR #define CFE_ES_RST_ACCESS_ERR ((CFE_Status_t)0xc4000015)
Reset Area Access Error.

Occurs when the BSP is not successful in returning the reset area address.

Definition at line 474 of file cfe_error.h.

10.11.2.38 CFE_ES_TASK_DELETE_ERR #define CFE_ES_TASK_DELETE_ERR (([CFE_Status_t](#))0xc4000026)
Task Delete Error.

Occurs when trying to delete a task that ES is cleaning up.

Definition at line 614 of file [cfe_error.h](#).

10.11.2.39 CFE_ES_TIMER_DELETE_ERR #define CFE_ES_TIMER_DELETE_ERR (([CFE_Status_t](#))0xc4000024)
Timer Delete Error.

Occurs when trying to delete a Timer that belongs to a task that ES is cleaning up.

Definition at line 597 of file [cfe_error.h](#).

10.11.2.40 CFE_EVS_APP_FILTER_OVERLOAD #define CFE_EVS_APP_FILTER_OVERLOAD (([CFE_Status_t](#))0xc2000004)
Application Filter Overload.

Number of Application event filters input upon registration is greater than [CFE_PLATFORM_EVS_MAX_EVENT_FILTERS](#)

Definition at line 276 of file [cfe_error.h](#).

10.11.2.41 CFE_EVS_APP_ILLEGAL_APP_ID #define CFE_EVS_APP_ILLEGAL_APP_ID (([CFE_Status_t](#))0xc2000003)
Illegal Application ID.

Application ID returned by [CFE_ES_GetAppIDByName](#) is greater than [CFE_PLATFORM_ES_MAX_APPLICATIONS](#)

Definition at line 267 of file [cfe_error.h](#).

10.11.2.42 CFE_EVS_APP_NOT_REGISTERED #define CFE_EVS_APP_NOT_REGISTERED (([CFE_Status_t](#))0xc2000002)
Application Not Registered.

Calling application never previously called [CFE_EVS_Register](#)

Definition at line 258 of file [cfe_error.h](#).

10.11.2.43 CFE_EVS_APP_SQUELCHED #define CFE_EVS_APP_SQUELCHED (([CFE_Status_t](#))0xc2000009)
Event squelched.

Event squelched due to being sent at too high a rate

Definition at line 318 of file [cfe_error.h](#).

10.11.2.44 CFE_EVS_EVT_NOT_REGISTERED #define CFE_EVS_EVT_NOT_REGISTERED (([CFE_Status_t](#))0xc2000006)
Event Not Registered.

[CFE_EVS_ResetFilter](#) EventID argument was not found in any event filter registered by the calling application.

Definition at line 294 of file [cfe_error.h](#).

10.11.2.45 CFE_EVS_FILE_WRITE_ERROR #define CFE_EVS_FILE_WRITE_ERROR (([CFE_Status_t](#))0xc2000007)
File Write Error.

A file write error occurred while processing an EVS command

Definition at line 302 of file [cfe_error.h](#).

10.11.2.46 CFE_EVS_INVALID_PARAMETER #define CFE_EVS_INVALID_PARAMETER (([CFE_Status_t](#))0xc2000008)
Invalid Pointer.

Invalid parameter supplied to EVS command

Definition at line 310 of file [cfe_error.h](#).

10.11.2.47 CFE_EVS_NOT_IMPLEMENTED #define CFE_EVS_NOT_IMPLEMENTED ((CFE_Status_t)0xc200ffff)
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.

Definition at line 329 of file cfe_error.h.

10.11.2.48 CFE_EVS_RESET_AREA_POINTER #define CFE_EVS_RESET_AREA_POINTER ((CFE_Status_t)0xc2000005)
Reset Area Pointer Failure.

Could not get pointer to the ES Reset area, so we could not get the pointer to the EVS Log.

Definition at line 285 of file cfe_error.h.

10.11.2.49 CFE_EVS_UNKNOWN_FILTER #define CFE_EVS_UNKNOWN_FILTER ((CFE_Status_t)0xc2000001)
Unknown Filter.

[CFE_EVS_Register](#) FilterScheme parameter was illegal

Definition at line 250 of file cfe_error.h.

10.11.2.50 CFE_FS_BAD_ARGUMENT #define CFE_FS_BAD_ARGUMENT ((CFE_Status_t)0xc6000001)
Bad Argument.

A parameter given by a caller to a File Services API did not pass validation checks.

Definition at line 692 of file cfe_error.h.

10.11.2.51 CFE_FS_FNAME_TOO_LONG #define CFE_FS_FNAME_TOO_LONG ((CFE_Status_t)0xc6000003)
Filename Too Long.

FS filename string is too long

Definition at line 708 of file cfe_error.h.

10.11.2.52 CFE_FS_INVALID_PATH #define CFE_FS_INVALID_PATH ((CFE_Status_t)0xc6000002)
Invalid Path.

FS was unable to extract a filename from a path string

Definition at line 700 of file cfe_error.h.

10.11.2.53 CFE_FS_NOT_IMPLEMENTED #define CFE_FS_NOT_IMPLEMENTED ((CFE_Status_t)0xc600ffff)
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.

Definition at line 719 of file cfe_error.h.

10.11.2.54 CFE_SB_BAD_ARGUMENT #define CFE_SB_BAD_ARGUMENT ((CFE_Status_t)0xca000003)
Bad Argument.

A parameter given by a caller to a Software Bus API did not pass validation checks.

Definition at line 750 of file cfe_error.h.

10.11.2.55 CFE_SB_BUF_ALOC_ERR #define CFE_SB_BUF_ALOC_ERR (([CFE_Status_t](#))0xca000008)

Buffer Allocation Error.

Returned 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.

Definition at line 808 of file [cfe_error.h](#).

10.11.2.56 CFE_SB_BUFFER_INVALID #define CFE_SB_BUFFER_INVALID (([CFE_Status_t](#))0xca00000e)

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.

Definition at line 859 of file [cfe_error.h](#).

10.11.2.57 CFE_SB_INTERNAL_ERR #define CFE_SB_INTERNAL_ERR (([CFE_Status_t](#))0xca00000c)

Internal Error.

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 840 of file [cfe_error.h](#).

10.11.2.58 CFE_SB_MAX_DESTS_MET #define CFE_SB_MAX_DESTS_MET (([CFE_Status_t](#))0xca00000a)

Max Destinations Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate 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 830 of file [cfe_error.h](#).

10.11.2.59 CFE_SB_MAX_MSGS_MET #define CFE_SB_MAX_MSGS_MET (([CFE_Status_t](#))0xca000009)

Max Messages Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another unique message ID because the platform configuration parameter [CFE_PLATFORM_SB_MAX_MSG_IDS](#) has been met.

Definition at line 818 of file [cfe_error.h](#).

10.11.2.60 CFE_SB_MAX_PIPES_MET #define CFE_SB_MAX_PIPES_MET (([CFE_Status_t](#))0xca000004)

Max Pipes Met.

This error code will be returned from [CFE_SB_CreatePipe](#) when the SB cannot accommodate 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 761 of file [cfe_error.h](#).

10.11.2.61 CFE_SB_MSG_TOO_BIG #define CFE_SB_MSG_TOO_BIG (([CFE_Status_t](#))0xca000007)

Message 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](#)

Definition at line 795 of file [cfe_error.h](#).

10.11.2.62 CFE_SB_NO_MESSAGE #define CFE_SB_NO_MESSAGE ((CFE_Status_t)0xca000002)

No Message.

When "Polling" a pipe for a message in [CFE_SB_ReceiveBuffer](#), this return value indicates that there was not a message on the pipe.

Definition at line 741 of file cfe_error.h.

10.11.2.63 CFE_SB_NOT_IMPLEMENTED #define CFE_SB_NOT_IMPLEMENTED ((CFE_Status_t)0xca00ffff)

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.

Definition at line 870 of file cfe_error.h.

10.11.2.64 CFE_SB_PIPE_CR_ERR #define CFE_SB_PIPE_CR_ERR ((CFE_Status_t)0xca000005)

Pipe Create Error.

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 772 of file cfe_error.h.

10.11.2.65 CFE_SB_PIPE_RD_ERR #define CFE_SB_PIPE_RD_ERR ((CFE_Status_t)0xca000006)

Pipe Read Error.

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 785 of file cfe_error.h.

10.11.2.66 CFE_SB_TIME_OUT #define CFE_SB_TIME_OUT ((CFE_Status_t)0xca000001)

Time Out.

In [CFE_SB_ReceiveBuffer](#), this return value indicates that a packet has not been received in the time given in the "timeout" parameter.

Definition at line 732 of file cfe_error.h.

10.11.2.67 CFE_SB_WRONG_MSG_TYPE #define CFE_SB_WRONG_MSG_TYPE ((CFE_Status_t)0xca00000d)

Wrong Message Type.

This error code will be returned when a request such as [CFE_MSG_SetMsgTime](#) is made on a packet that does not include a field for msg time.

Definition at line 849 of file cfe_error.h.

10.11.2.68 CFE_STATUS_BAD_COMMAND_CODE #define CFE_STATUS_BAD_COMMAND_CODE ((CFE_Status_t)0xc8000004)

Bad Command Code.

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 182 of file cfe_error.h.

10.11.2.69 CFE_STATUS_EXTERNAL_RESOURCE_FAIL #define CFE_STATUS_EXTERNAL_RESOURCE_FA←
IL ((CFE_Status_t) 0xc8000005)

External failure.

This error indicates that the operation failed for some reason outside the scope of CFE. The real failure may have been in OSAL, PSP, or another dependent library.

Details of the original failure should be written to syslog and/or a system event before returning this error.

Definition at line 194 of file cfe_error.h.

10.11.2.70 CFE_STATUS_INCORRECT_STATE #define CFE_STATUS_INCORRECT_STATE ((int32) 0xc8000009)

Cannot process request at this time.

The system is not currently in the correct state to accept the request at this time.

Definition at line 227 of file cfe_error.h.

10.11.2.71 CFE_STATUS_NO_COUNTER_INCREMENT #define CFE_STATUS_NO_COUNTER_INCREMENT ((CFE_Status_t) 0x48000000)

No Counter Increment.

Informational code indicating that a command was processed successfully but that the command counter should *not* be incremented.

Definition at line 155 of file cfe_error.h.

10.11.2.72 CFE_STATUS_NOT_IMPLEMENTED #define CFE_STATUS_NOT_IMPLEMENTED ((CFE_Status_t) 0xc800ffff)

Not Implemented.

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 238 of file cfe_error.h.

10.11.2.73 CFE_STATUS_RANGE_ERROR #define CFE_STATUS_RANGE_ERROR ((int32) 0xc8000008)

Request or input value is out of range.

A message, table, or function call input contained a value that was outside the acceptable range, and the request was rejected.

Definition at line 219 of file cfe_error.h.

10.11.2.74 CFE_STATUS_REQUEST_ALREADY_PENDING #define CFE_STATUS_REQUEST_ALREADY_PENDI←
NG ((int32) 0xc8000006)

Request already pending.

Commands or requests are already pending or the pending request limit has been reached. No more requests can be made until the current request(s) complete.

Definition at line 203 of file cfe_error.h.

10.11.2.75 CFE_STATUS_UNKNOWN_MSG_ID #define CFE_STATUS_UNKNOWN_MSG_ID ((CFE_Status_t) 0xc8000003)

Unknown Message ID.

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 173 of file cfe_error.h.

10.11.2.76 CFE_STATUS_VALIDATION_FAILURE #define CFE_STATUS_VALIDATION_FAILURE ((int32)0xc8000007)
Request or input value failed basic structural validation.

A message or table input was not in the proper format to be understood and processed by an application, and was rejected.

Definition at line 211 of file cfe_error.h.

10.11.2.77 CFE_STATUS_WRONG_MSG_LENGTH #define CFE_STATUS_WRONG_MSG_LENGTH ((CFE_Status_t)0xc8000002)
Wrong Message Length.

This error code will be returned when a message validation process determined that the message length is incorrect

Definition at line 164 of file cfe_error.h.

10.11.2.78 CFE_SUCCESS #define CFE_SUCCESS ((CFE_Status_t)0)
Successful execution.

Operation was performed successfully

Definition at line 147 of file cfe_error.h.

10.11.2.79 CFE_TBL_BAD_ARGUMENT #define CFE_TBL_BAD_ARGUMENT ((CFE_Status_t)0xcc00002d)
Bad Argument.

A parameter given by a caller to a Table API did not pass validation checks.

Definition at line 1281 of file cfe_error.h.

10.11.2.80 CFE_TBL_ERR_ACCESS #define CFE_TBL_ERR_ACCESS ((CFE_Status_t)0xcc00002c)
Error code indicating that the TBL file could not be opened by the OS.

Definition at line 1272 of file cfe_error.h.

10.11.2.81 CFE_TBL_ERR_BAD_CONTENT_ID #define CFE_TBL_ERR_BAD_CONTENT_ID ((CFE_Status_t)0xcc000016)
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.

Definition at line 1064 of file cfe_error.h.

10.11.2.82 CFE_TBL_ERR_BAD_PROCESSOR_ID #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((CFE_Status_t)0xcc000029)
Bad Processor ID.

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 1252 of file cfe_error.h.

10.11.2.83 CFE_TBL_ERR_BAD_SPACECRAFT_ID #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((CFE_Status_t)0xcc000028)
Bad Spacecraft ID.

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 1241 of file cfe_error.h.

10.11.2.84 CFE_TBL_ERR_BAD_SUBTYPE_ID #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((CFE_Status_t)0xcc00001B)
Bad Subtype ID.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.
Definition at line 1105 of file cfe_error.h.

10.11.2.85 CFE_TBL_ERR_DUMP_ONLY #define CFE_TBL_ERR_DUMP_ONLY ((CFE_Status_t)0xcc000010)
Dump Only Error.

The calling Application has attempted to perform a load on a table that was created with "Dump Only" attributes.
Definition at line 1016 of file cfe_error.h.

10.11.2.86 CFE_TBL_ERR_DUPLICATE_DIFF_SIZE #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((CFE_Status_t)0xcc00000C)
Duplicate Table With Different 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.

Definition at line 977 of file cfe_error.h.

10.11.2.87 CFE_TBL_ERR_DUPLICATE_NOT OWNED #define CFE_TBL_ERR_DUPLICATE_NOT OWNED ((CFE_Status_t)0xcc00000D)
Duplicate Table And 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.

Definition at line 987 of file cfe_error.h.

10.11.2.88 CFE_TBL_ERR_FILE_FOR_WRONG_TABLE #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc00000E)
File For Wrong Table.

The calling Application tried to load a table using a file whose header indicated that it was for a different table.
Definition at line 1149 of file cfe_error.h.

10.11.2.89 CFE_TBL_ERR_FILE_SIZE_INCONSISTENT #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((CFE_Status_t)0xcc00001C)
File Size Inconsistent.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.
Definition at line 1114 of file cfe_error.h.

10.11.2.90 CFE_TBL_ERR_FILE_TOO_LARGE #define CFE_TBL_ERR_FILE_TOO_LARGE ((CFE_Status_t)0xcc000014)
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.

Definition at line 1044 of file cfe_error.h.

10.11.2.91 CFE_TBL_ERR_FILENAME_TOO_LONG #define CFE_TBL_ERR_FILENAME_TOO_LONG ((CFE_Status_t)0xcc00001F)
Filename Too Long.

The calling Application tried to load a table using a filename that was too long.
Definition at line 1140 of file cfe_error.h.

10.11.2.92 CFE_TBL_ERR_HANDLES_FULL #define CFE_TBL_ERR_HANDLES_FULL ((CFE_Status_t)0xcc00000B)
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.

Definition at line 967 of file cfe_error.h.

10.11.2.93 CFE_TBL_ERR_ILLEGAL_SRC_TYPE #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((CFE_Status_t)0xcc000011)
Illegal Source Type.

The calling Application called [CFE_TBL_Load](#) with an illegal value for the second parameter.

Definition at line 1025 of file cfe_error.h.

10.11.2.94 CFE_TBL_ERR_INVALID_HANDLE #define CFE_TBL_ERR_INVALID_HANDLE ((CFE_Status_t)0xcc000001)
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.

Definition at line 884 of file cfe_error.h.

10.11.2.95 CFE_TBL_ERR_INVALID_NAME #define CFE_TBL_ERR_INVALID_NAME ((CFE_Status_t)0xcc000002)
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.

Definition at line 894 of file cfe_error.h.

10.11.2.96 CFE_TBL_ERR_INVALID_OPTIONS #define CFE_TBL_ERR_INVALID_OPTIONS ((CFE_Status_t)0xcc000025)
Invalid Options.

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 1206 of file cfe_error.h.

10.11.2.97 CFE_TBL_ERR_INVALID_SIZE #define CFE_TBL_ERR_INVALID_SIZE ((CFE_Status_t)0xcc000003)
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

Definition at line 905 of file cfe_error.h.

10.11.2.98 CFE_TBL_ERR_LOAD_IN_PROGRESS #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((CFE_Status_t)0xcc000012)
Load In Progress.

The calling Application called [CFE_TBL_Load](#) when another Application was trying to load the table.

Definition at line 1034 of file cfe_error.h.

10.11.2.99 CFE_TBL_ERR_LOAD_INCOMPLETE #define CFE_TBL_ERR_LOAD_INCOMPLETE ((CFE_Status_t)0xcc000021)
Load Incomplete.

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 1158 of file cfe_error.h.

10.11.2.100 CFE_TBL_ERR_NEVER_LOADED #define CFE_TBL_ERR_NEVER_LOADED ((CFE_Status_t)0xcc000005)
Never Loaded.

Table has not been loaded with data.

Definition at line 921 of file cfe_error.h.

10.11.2.101 CFE_TBL_ERR_NO_ACCESS #define CFE_TBL_ERR_NO_ACCESS ((CFE_Status_t)0xcc000008)
No Access.

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 949 of file cfe_error.h.

10.11.2.102 CFE_TBL_ERR_NO_BUFFER_AVAIL #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((CFE_Status_t)0xcc00000F)
No Buffer Available.

The calling Application has tried to allocate a working buffer but none were available.

Definition at line 1007 of file cfe_error.h.

10.11.2.103 CFE_TBL_ERR_NO_STD_HEADER #define CFE_TBL_ERR_NO_STD_HEADER ((CFE_Status_t)0xcc00001D)
No Standard Header.

The calling Application tried to access a table file whose standard cFE File Header was the wrong size, etc.

Definition at line 1122 of file cfe_error.h.

10.11.2.104 CFE_TBL_ERR_NO_TBL_HEADER #define CFE_TBL_ERR_NO_TBL_HEADER ((CFE_Status_t)0xcc00001E)
No Table Header.

The calling Application tried to access a table file whose standard cFE Table File Header was the wrong size, etc.

Definition at line 1131 of file cfe_error.h.

10.11.2.105 CFE_TBL_ERR_PARTIAL_LOAD #define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)
Partial Load Error.

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 1180 of file cfe_error.h.

10.11.2.106 CFE_TBL_ERR_REGISTRY_FULL #define CFE_TBL_ERR_REGISTRY_FULL (([CFE_Status_t](#))0xcc000006)
Registry Full.

An application attempted to create a table and the Table registry already contained [CFE_PLATFORM_TBL_MAX_NUM_TABLES](#) in it.

Definition at line 930 of file [cfe_error.h](#).

10.11.2.107 CFE_TBL_ERR_SHORT_FILE #define CFE_TBL_ERR_SHORT_FILE (([CFE_Status_t](#))0xcc00002b)

Error code indicating that the TBL file is shorter than indicated in the file header.

Definition at line 1266 of file [cfe_error.h](#).

10.11.2.108 CFE_TBL_ERR_UNREGISTERED #define CFE_TBL_ERR_UNREGISTERED (([CFE_Status_t](#))0xcc000009)
Unregistered.

The calling application is trying to access a table that has been unregistered.

Definition at line 958 of file [cfe_error.h](#).

10.11.2.109 CFE_TBL_INFO_DUMP_PENDING #define CFE_TBL_INFO_DUMP_PENDING (([CFE_Status_t](#))0x4c000024)
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.

Definition at line 1190 of file [cfe_error.h](#).

10.11.2.110 CFE_TBL_INFO_NO_UPDATE_PENDING #define CFE_TBL_INFO_NO_UPDATE_PENDING (([CFE_Status_t](#))0x4c000017)
No Update Pending.

The calling Application has attempted to update a table without a pending load.

Definition at line 1072 of file [cfe_error.h](#).

10.11.2.111 CFE_TBL_INFO_NO_VALIDATION_PENDING #define CFE_TBL_INFO_NO_VALIDATION_PENDI←
NG (([CFE_Status_t](#))0x4c00001A)

No Validation Pending

The calling Application tried to validate a table that did not have a validation request pending.

Definition at line 1096 of file [cfe_error.h](#).

10.11.2.112 CFE_TBL_INFO_RECOVERED_TBL #define CFE_TBL_INFO_RECOVERED_TBL (([CFE_Status_t](#))0x4c000027)
Recovered Table.

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 1230 of file [cfe_error.h](#).

10.11.2.113 CFE_TBL_INFO_TABLE_LOCKED #define CFE_TBL_INFO_TABLE_LOCKED (([CFE_Status_t](#))0x4c000018)
Table Locked.

The calling Application tried to update a table that is locked by another user.

Definition at line 1080 of file [cfe_error.h](#).

10.11.2.114 CFE_TBL_INFO_UPDATE_PENDING #define CFE_TBL_INFO_UPDATE_PENDING (([CFE_Status_t](#))0x4c000004)
Update Pending.

The calling Application has identified a table that has a load pending.

Definition at line 913 of file cfe_error.h.

10.11.2.115 CFE_TBL_INFO_UPDATED #define CFE_TBL_INFO_UPDATED (([CFE_Status_t](#))0x4c00000E)
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.

Definition at line 998 of file cfe_error.h.

10.11.2.116 CFE_TBL_INFO_VALIDATION_PENDING #define CFE_TBL_INFO_VALIDATION_PENDING (([CFE_Status_t](#))0x4c000010)
Validation Pending

The calling Application should call [CFE_TBL_Validate](#) for the specified table.

Definition at line 1088 of file cfe_error.h.

10.11.2.117 CFE_TBL_MESSAGE_ERROR #define CFE_TBL_MESSAGE_ERROR (([CFE_Status_t](#))0xcc00002a)
Message Error.

Error code indicating that the TBL command was not processed successfully and that the error counter should be incremented.

Definition at line 1260 of file cfe_error.h.

10.11.2.118 CFE_TBL_NOT_IMPLEMENTED #define CFE_TBL_NOT_IMPLEMENTED (([CFE_Status_t](#))0xcc00ffff)
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.

Definition at line 1292 of file cfe_error.h.

10.11.2.119 CFE_TBL_WARN_DUPLICATE #define CFE_TBL_WARN_DUPLICATE (([CFE_Status_t](#))0x4c000007)
Duplicate Warning.

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 940 of file cfe_error.h.

10.11.2.120 CFE_TBL_WARN_NOT_CRITICAL #define CFE_TBL_WARN_NOT_CRITICAL (([CFE_Status_t](#))0x4c000026)
Not Critical Warning.

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 1217 of file cfe_error.h.

10.11.2.121 CFE_TBL_WARN_PARTIAL_LOAD #define CFE_TBL_WARN_PARTIAL_LOAD (([CFE_Status_t](#))0x4c000022)
Partial Load Warning.

The calling Application tried to load a table file whose header claimed the load did not start with the first byte. It should be noted that [CFE_TBL_WARN_SHORT_FILE](#) also indicates a partial load.

Definition at line 1168 of file [cfe_error.h](#).

10.11.2.122 CFE_TBL_WARN_SHORT_FILE #define CFE_TBL_WARN_SHORT_FILE (([CFE_Status_t](#))0x4c000015)
Short File Warning.

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 1055 of file [cfe_error.h](#).

10.11.2.123 CFE_TIME_BAD_ARGUMENT #define CFE_TIME_BAD_ARGUMENT (([CFE_Status_t](#))0xce000005)
Bad Argument.

A parameter given by a caller to a TIME Services API did not pass validation checks.

Definition at line 1364 of file [cfe_error.h](#).

10.11.2.124 CFE_TIME_CALLBACK_NOT_REGISTERED #define CFE_TIME_CALLBACK_NOT_REGISTERED (([CFE_Status_t](#))0xce000006)
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.

Definition at line 1355 of file [cfe_error.h](#).

10.11.2.125 CFE_TIME_INTERNAL_ONLY #define CFE_TIME_INTERNAL_ONLY (([CFE_Status_t](#))0xce000001)
Internal Only.

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 1319 of file [cfe_error.h](#).

10.11.2.126 CFE_TIME_NOT_IMPLEMENTED #define CFE_TIME_NOT_IMPLEMENTED (([CFE_Status_t](#))0xce00ffff)
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.

Definition at line 1307 of file [cfe_error.h](#).

10.11.2.127 CFE_TIME_OUT_OF_RANGE #define CFE_TIME_OUT_OF_RANGE (([CFE_Status_t](#))0xce000002)
Out Of Range.

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](#).

10.11.2.128 CFE_TIME_TOO_MANY_SYNCH_CALLBACKS #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS
KS ((CFE_Status_t)0xce000003)

Too Many Sync 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.

Definition at line 1345 of file cfe_error.h.

10.12 cFE Resource ID APIs

Functions

- `CFE_Status_t CFE_ES_AppID_ToIndex (CFE_ES_AppId_t AppID, uint32 *Idx)`
Obtain an index value correlating to an ES Application ID.
- `int32 CFE_ES_LibID_ToIndex (CFE_ES_LibId_t LibId, uint32 *Idx)`
Obtain an index value correlating to an ES Library ID.
- `CFE_Status_t CFE_ES_TaskID_ToIndex (CFE_ES_TaskId_t TaskID, uint32 *Idx)`
Obtain an index value correlating to an ES Task ID.
- `CFE_Status_t CFE_ES_CounterID_ToIndex (CFE_ES_CounterId_t CounterId, uint32 *Idx)`
Obtain an index value correlating to an ES Counter ID.

10.12.1 Detailed Description

10.12.2 Function Documentation

10.12.2.1 CFE_ES_AppID_ToIndex() `CFE_Status_t CFE_ES_AppID_ToIndex (`
`CFE_ES_AppId_t AppID,`
`uint32 * Idx)`

Obtain an index value correlating to an ES Application ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of an application and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

Note

There is no inverse of this function - indices cannot be converted back to the original AppID value. The caller should retain the original ID for future use.

Parameters

in	<i>AppID</i>	Application ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.

10.12.2.2 CFE_ES_CounterID_ToIndex() `CFE_Status_t CFE_ES_CounterID_ToIndex (`
`CFE_ES_CounterId_t CounterId,`
`uint32 * Idx)`

Obtain an index value correlating to an ES Counter ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Counter IDs will never overlap, but the index of a Counter and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

Note

There is no inverse of this function - indices cannot be converted back to the original CounterID value. The caller should retain the original ID for future use.

Parameters

in	<i>Counter</i> <i>Id</i>	Counter ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

10.12.2.3 CFE_ES_LibID_ToIndex() `int32 CFE_ES_LibID_ToIndex (`
 `CFE_ES_LibId_t LibId,`
 `uint32 * Idx)`

Obtain an index value correlating to an ES Library ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Library IDs will never overlap, but the index of an Library and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

Note

There is no inverse of this function - indices cannot be converted back to the original LibID value. The caller should retain the original ID for future use.

Parameters

in	<i>Lib</i> <i>Id</i>	Library ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

```
10.12.2.4 CFE_ES_TaskID_ToIndex() CFE_Status_t CFE_ES_TaskID_ToIndex (
    CFE_ES_TaskId_t TaskID,
    uint32 * Idx )
```

Obtain an index value correlating to an ES Task ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Task IDs will never overlap, but the index of a Task and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

Note

There is no inverse of this function - indices cannot be converted back to the original TaskID value. The caller should retain the original ID for future use.

Parameters

in	<i>TaskID</i>	Task ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

10.13 cFE Entry/Exit APIs

Functions

- void [CFE_ES_Main](#) (uint32 StartType, uint32 StartSubtype, uint32 ModelId, const char *StartFilePath)
cFE Main Entry Point used by Board Support Package to start cFE
- [CFE_Status_t CFE_ES_ResetCFE](#) (uint32 ResetType)
Reset the cFE Core and all cFE Applications.

10.13.1 Detailed Description

10.13.2 Function Documentation

10.13.2.1 CFE_ES_Main() void CFE_ES_Main (

```
    uint32 StartType,
    uint32 StartSubtype,
    uint32 ModelId,
    const char * StartFilePath )
```

cFE Main Entry Point used by Board Support Package to start cFE

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	<i>StartType</i>	Identifies whether this was a CFE_PSP_RST_TYPE_POWERON or CFE_PSP_RST_TYPE_PROCESSOR .
in	<i>StartSubtype</i>	Specifies, in more detail, what caused the <i>StartType</i> identified above. See CFE_PSP_RST_SUBTYPE_POWER_CYCLE for possible examples.
in	<i>ModelId</i>	Identifies the source of the Boot as determined by the BSP.
in	<i>StartFilePath</i>	Identifies the startup file to use to initialize the cFE apps.

See also

[CFE_ES_ResetCFE](#)

10.13.2.2 CFE_ES_ResetCFE() [CFE_Status_t CFE_ES_ResetCFE](#) (

```
    uint32 ResetType )
```

Reset the cFE Core and all cFE Applications.

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	<i>ResetType</i>	Identifies the type of reset desired. Allowable settings are: <ul style="list-style-type: none">• 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.
----	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NOT_IMPLEMENTED	Not Implemented.

See also

[CFE_ES_Main](#)

10.14 cFE Application Control APIs

Functions

- [CFE_Status_t CFE_ES_RestartApp \(CFE_ES_AppId_t AppID\)](#)
Restart a single cFE Application.
- [CFE_Status_t CFE_ES_ReloadApp \(CFE_ES_AppId_t AppID, const char *AppFileName\)](#)
Reload a single cFE Application.
- [CFE_Status_t CFE_ES_DeleteApp \(CFE_ES_AppId_t AppID\)](#)
Delete a cFE Application.

10.14.1 Detailed Description

10.14.2 Function Documentation

10.14.2.1 CFE_ES_DeleteApp() [CFE_Status_t CFE_ES_DeleteApp \(CFE_ES_AppId_t AppID \)](#)

Delete a cFE Application.

Description

This API causes a cFE Application to be stopped deleted.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>AppID</i>	Identifies the application to be reset.
----	--------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_SUCCESS	Successful execution.

See also

[CFE_ES_RestartApp](#), [CFE_ES_ReloadApp](#)

10.14.2.2 CFE_ES_ReloadApp() [CFE_Status_t CFE_ES_ReloadApp \(CFE_ES_AppId_t AppID, const char * AppFileName \)](#)

Reload a single cFE Application.

Description

This API causes a cFE Application to be stopped and restarted from the specified file.

Assumptions, External Events, and Notes:

The filename is checked for existence prior to load. A missing file will be reported and the reload operation will be aborted prior to unloading the app.

Goes through the standard CFE_ES_CleanUpApp which unloads, then attempts a load using the specified file name. 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 started by loading the application via the ES_STARTAPP command ([CFE_ES_START_APP_CC](#)).

Parameters

in	<i>AppID</i>	Identifies the application to be reset.
in	<i>AppFileName</i>	Identifies the new file to start (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_SUCCESS	Successful execution.
CFE_ES_FILE_IO_ERR	File IO Error.

See also

[CFE_ES_RestartApp](#), [CFE_ES_DeleteApp](#), [CFE_ES_START_APP_CC](#)

10.14.2.3 CFE_ES_RestartApp() [CFE_Status_t](#) CFE_ES_RestartApp ([CFE_ES_AppId_t](#) *AppID*)

Restart a single cFE Application.

Description

This API causes a cFE Application to be unloaded and restarted from the same file name as the last start.

Assumptions, External Events, and Notes:

The filename is checked for existence prior to load. A missing file will be reported and the reload operation will be aborted prior to unloading the app.

Goes through the standard CFE_ES_CleanUpApp which unloads, then attempts a load using the original file name. In the event that an application cannot be reloaded due to a missing file or any other load issue, the application may no longer be restarted or reloaded when given a valid load file (the app has been deleted and no longer exists). To recover, the application may be started by loading the application via the ES_STARTAPP command ([CFE_ES_START_APP_CC](#)).

Parameters

in	<i>AppID</i>	Identifies the application to be reset.
----	--------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_FILE_IO_ERR</i>	File IO Error.
<i>CFE_SUCCESS</i>	Successful execution.

See also

[CFE_ES_ReloadApp](#), [CFE_ES_DeleteApp](#)

10.15 cFE Application Behavior APIs

Functions

- void [CFE_ES_ExitApp](#) (`uint32 ExitStatus`)
Exit a cFE Application.
- bool [CFE_ES_RunLoop](#) (`uint32 *RunStatus`)
Check for Exit, Restart, or Reload commands.
- [CFE_Status_t 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.
- void [CFE_ES_IncrementTaskCounter](#) (`void`)
Increments the execution counter for the calling task.

10.15.1 Detailed Description

10.15.2 Function Documentation

10.15.2.1 [CFE_ES_ExitApp\(\)](#) `void CFE_ES_ExitApp (` `uint32 ExitStatus)`

Exit a cFE Application.

Description

This API is the "Exit Point" for the cFE application

Assumptions, External Events, and Notes:

None

Parameters

<code>in</code>	<code>ExitStatus</code>	Acceptable values are: <ul style="list-style-type: none">• 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_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.
-----------------	-------------------------	--

See also

[CFE_ES_RunLoop](#)

Referenced by CF_AppMain().

```
10.15.2.2 CFE_ES_IncrementTaskCounter() void CFE_ES_IncrementTaskCounter (
    void )
```

Increments the execution counter for the calling task.

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 Applications that call the CFE_ES_RunLoop call.

See also

[CFE_ES_RunLoop](#)

```
10.15.2.3 CFE_ES_RunLoop() bool CFE_ES_RunLoop (
    uint32 * RunStatus )
```

Check for Exit, Restart, or Reload commands.

Description

This is the API that allows an app to check for exit requests from the system, or request shutdown from the system.

Assumptions, External Events, and Notes:

This API updates the internal task counter tracked by ES for the calling task. For ES to report application counters correctly this API should be called from the main app task as part of it's main processing loop.

In the event of a externally initiated app shutdown request (such as the APP_STOP, APP_RELOAD, and APP_RESET commands) or if a system error occurs requiring the app to be shut down administratively, this function returns "false" and optionally sets the "RunStatus" output to further indicate the specific application state.

If "RunStatus" is passed as non-NULL, it should point to a local status variable containing the requested status to ES. Normally, this should be initialized to [CFE_ES_RunStatus_APP_RUN](#) during application start up, and should remain as this value during normal operation.

If "RunStatus" is set to [CFE_ES_RunStatus_APP_EXIT](#) or [CFE_ES_RunStatus_APP_ERROR](#) on input, this acts as a shutdown request - [CFE_ES_RunLoop\(\)](#) function will return "false", and a shutdown will be initiated similar to if ES had been externally commanded to shut down the app.

If "RunStatus" is not used, it should be passed as NULL. In this mode, only the boolean return value is relevant, which will indicate if an externally-initiated shutdown request is pending.

Parameters

in, out	<i>RunStatus</i>	Optional pointer to a variable containing the desired run status
---------	------------------	--

Returns

Boolean indicating application should continue running

Return values

<i>true</i>	Application should continue running
<i>false</i>	Application should not continue running

See also

[CFE_ES_ExitApp](#)

Referenced by CF_AppMain().

10.15.2.4 CFE_ES_WaitForStartupSync() `void CFE_ES_WaitForStartupSync (`
`uint32 TimeOutMilliseconds)`

Allow an Application to Wait for the "OPERATIONAL" global system state.

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

<i>in</i>	<i>TimeOutMilliseconds</i>	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](#)

10.15.2.5 CFE_ES_WaitForSystemState() `CFE_Status_t CFE_ES_WaitForSystemState (`
`uint32 MinSystemState,`
`uint32 TimeOutMilliseconds)`

Allow an Application to Wait for a minimum global system state.

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 [CFE_ES_WaitForStartupSync](#)

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	<i>MinSystemState</i>	Determine the state of the App
in	<i>TimeOutMilliseconds</i>	The timeout value in Milliseconds. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	State successfully achieved
<i>CFE_ES_OPERATION_TIMED_OUT</i>	(return value only verified in coverage test) Timeout was reached

See also

[CFE_ES_RunLoop](#)

10.16 cFE Information APIs

Functions

- `int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)`
Return the most recent Reset Type.
- `CFE_Status_t CFE_ES_GetAppID (CFE_ES_AppId_t *AppIdPtr)`
Get an Application ID for the calling Application.
- `CFE_Status_t CFE_ES_GetTaskID (CFE_ES_TaskId_t *TaskIdPtr)`
Get the task ID of the calling context.
- `CFE_Status_t CFE_ES_GetAppIDByName (CFE_ES_AppId_t *AppIdPtr, const char *AppName)`
Get an Application ID associated with a specified Application name.
- `CFE_Status_t CFE_ES_GetLibIDByName (CFE_ES_LibId_t *LibIdPtr, const char *LibName)`
Get a Library ID associated with a specified Library name.
- `CFE_Status_t CFE_ES_GetAppName (char *AppName, CFE_ES_AppId_t AppId, size_t BufferLength)`
Get an Application name for a specified Application ID.
- `CFE_Status_t CFE_ES_GetLibName (char *LibName, CFE_ES_LibId_t LibId, size_t BufferLength)`
Get a Library name for a specified Library ID.
- `CFE_Status_t CFE_ES_GetAppInfo (CFE_ES_AppInfo_t *AppInfo, CFE_ES_AppId_t AppId)`
Get Application Information given a specified App ID.
- `CFE_Status_t CFE_ES_GetTaskInfo (CFE_ES_TaskInfo_t *TaskInfo, CFE_ES_TaskId_t TaskId)`
Get Task Information given a specified Task ID.
- `int32 CFE_ES_GetLibInfo (CFE_ES_AppInfo_t *LibInfo, CFE_ES_LibId_t LibId)`
Get Library Information given a specified Resource ID.
- `int32 CFE_ES_GetModuleInfo (CFE_ES_AppInfo_t *ModuleInfo, CFE_Resourceld_t Resourceld)`
Get Information given a specified Resource ID.

10.16.1 Detailed Description

10.16.2 Function Documentation

10.16.2.1 CFE_ES_GetAppID() `CFE_Status_t CFE_ES_GetAppID (`
`CFE_ES_AppId_t * AppIdPtr)`

Get an Application ID for the calling Application.

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

<code>out</code>	<code>AppIdPtr</code>	Pointer to variable that is to receive the Application's ID (must not be null). <code>*AppIdPtr</code> will be set to the application ID of the calling Application.
------------------	-----------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetResetType](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppName](#), [CFE_ES_GetTaskInfo](#)

10.16.2.2 CFE_ES_GetAppIDByName() *CFE_Status_t* CFE_ES_GetAppIDByName (

```
    CFE_ES_AppId_t * AppIdPtr,  
    const char * AppName )
```

Get an Application ID associated with a specified Application name.

Description

This routine retrieves the cFE Application ID associated with a specified Application name.

Assumptions, External Events, and Notes:

None

Parameters

<i>out</i>	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null).
<i>in</i>	<i>AppName</i>	Pointer to null terminated character string containing an Application name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_NAME_NOT_FOUND</i>	Resource Name Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetAppID](#), [CFE_ES_GetAppName](#), [CFE_ES_GetAppInfo](#)

10.16.2.3 CFE_ES_GetAppInfo() *CFE_Status_t* CFE_ES_GetAppInfo (

```
    CFE_ES_AppInfo_t * AppInfo,
```

`CFE_ES_AppId_t AppId)`

Get Application Information given a specified App ID.

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

out	<i>AppInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>AppId</i>	ID of application to obtain information about

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

See also

[CFE_ES_GetAppID](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppName](#)

10.16.2.4 CFE_ES_GetAppName()

```
CFE_Status_t CFE_ES_GetAppName (
    char * AppName,
    CFE_ES_AppId_t AppId,
    size_t BufferLength )
```

Get an Application name for a specified Application ID.

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_RESOURCEID_NOT_VALID`), an empty string is returned.

Parameters

out	<i>AppName</i>	Pointer to a character array (must not be null) of at least <code>BufferLength</code> in size that will be filled with the appropriate Application name.
in	<i>AppId</i>	Application ID of Application whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <code>AppName</code> buffer. This routine will truncate the name to this length, if necessary.
<small>Generated by Doxygen</small>		

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetAppID](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppInfo](#)

10.16.2.5 CFE_ES_GetLibIDByName() [*CFE_Status_t*](#) CFE_ES_GetLibIDByName (
 [*CFE_ES_LibId_t*](#) * *LibIdPtr*,
 const char * *LibName*)

Get a Library ID associated with a specified Library name.

Description

This routine retrieves the cFE Library ID associated with a specified Library name.

Assumptions, External Events, and Notes:

None

Parameters

<i>out</i>	<i>LibIdPtr</i>	Pointer to variable that is to receive the Library's ID (must not be null).
<i>in</i>	<i>LibName</i>	Pointer to null terminated character string containing a Library name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_NAME_NOT_FOUND</i>	Resource Name Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetLibName](#)

10.16.2.6 CFE_ES_GetLibInfo() [*int32*](#) CFE_ES_GetLibInfo (
 [*CFE_ES_AppInfo_t*](#) * *LibInfo*,

```
CFE_ES_LibId_t LibId )
```

Get Library Information given a specified Resource ID.

Description

This routine retrieves the information about a Library associated with a specified ID. The information includes all of the information ES maintains for this resource type (documented in the CFE_ES_AppInfo_t type).

This shares the same output structure as CFE_ES_GetAppInfo, such that informational commands can be executed against either applications or libraries. When applied to a library, the task information in the structure will be omitted, as libraries do not have tasks associated.

Assumptions, External Events, and Notes:

None

Parameters

out	<i>LibInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>LibId</i>	ID of application to obtain information about

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetLibIDByName](#), [CFE_ES_GetLibName](#)

```
10.16.2.7 CFE_ES_GetLibName() CFE_Status_t CFE_ES_GetLibName (
    char * LibName,
    CFE_ES_LibId_t LibId,
    size_t BufferLength )
```

Get a Library name for a specified Library ID.

Description

This routine retrieves the cFE Library name associated with a specified Library ID.

Assumptions, External Events, and Notes:

In the case of a failure ([CFE_ES_ERR_RESOURCEID_NOT_VALID](#)), an empty string is returned.

Parameters

<i>out</i>	<i>LibName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Library name.
<i>in</i>	<i>LibId</i>	Library ID of Library whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters (must not be zero), including the null terminator, that can be put into the <i>LibName</i> buffer. This routine will truncate the name to this length, if necessary.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetLibIDByName](#)

10.16.2.8 CFE_ES_GetModuleInfo() `int32 CFE_ES_GetModuleInfo (CFE_ES_AppInfo_t * ModuleInfo, CFE_ResourceId_t ResourceId)`

Get Information given a specified Resource ID.

Description

This routine retrieves the information about an Application or Library associated with a specified ID.

This is a wrapper API that in turn calls either CFE_ES_GetAppInfo or CFE_ES_GetLibInfo if passed an AppId or LibId, respectively.

This allows commands originally targeted to operate on AppIDs to be easily ported to operate on either Libraries or Applications, where relevant.

Assumptions, External Events, and Notes:

None

Parameters

<i>out</i>	<i>ModuleInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
<i>in</i>	<i>ResourceId</i>	ID of application or library to obtain information about

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_GetLibInfo](#), [CFE_ES_GetApplInfo](#)

10.16.2.9 CFE_ES_GetResetType()

```
int32 CFE_ES_GetResetType (
    uint32 * ResetSubtypePtr )
```

Return the most recent Reset Type.

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, out	<i>ResetSubtypePtr</i>	Pointer to <code>uint32</code> 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. <i>ResetSubtypePtr</i> 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 ".
---------	------------------------	---

Returns

Processor reset type

Return values

CFE_PSP_RST_TYPE_POWERON	
CFE_PSP_RST_TYPE_PROCESSOR	

See also

[CFE_ES_GetAppID](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppName](#), [CFE_ES_GetTaskInfo](#)

10.16.2.10 CFE_ES_GetTaskID() `CFE_Status_t CFE_ES_GetTaskID (`
 `CFE_ES_TaskId_t * TaskIdPtr)`

Get the task ID of the calling context.

Description

This retrieves the current task context from OSAL

Assumptions, External Events, and Notes:

Applications which desire to call other CFE ES services such as CFE_ES_TaskGetInfo() should use this API rather than getting the ID from OSAL directly via [OS_TaskGetId\(\)](#).

Parameters

out	<i>TaskIdPtr</i>	Pointer to variable that is to receive the ID (must not be null). Will be set to the ID of the calling task.
-----	------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

10.16.2.11 CFE_ES_GetTaskInfo() `CFE_Status_t CFE_ES_GetTaskInfo (`
 `CFE_ES_TaskInfo_t * TaskInfo,`
 `CFE_ES_TaskId_t TaskId)`

Get Task Information given a specified Task ID.

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

out	<i>TaskInfo</i>	Pointer to a <code>CFE_ES_TaskInfo_t</code> structure (must not be null) that holds the specific task information. <code>*TaskInfo</code> is the filled out <code>CFE_ES_TaskInfo_t</code> structure containing the Task Name, Parent App Name, Parent App ID among other fields.
in	<i>TaskId</i>	Application ID of Application whose name is being requested.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetTaskID](#), [CFE_ES_GetTaskIDByName](#), [CFE_ES_GetTaskName](#)

10.17 cFE Child Task APIs

Functions

- `CFE_Status_t CFE_ES_CreateChildTask (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr FunctionPtr, CFE_ES_StackPointer_t StackPtr, size_t StackSize, CFE_ES_TaskPriority_Atom_t Priority, uint32 Flags)`

Creates a new task under an existing Application.
- `CFE_Status_t CFE_ES_GetTaskIDByName (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName)`

Get a Task ID associated with a specified Task name.
- `CFE_Status_t CFE_ES_GetTaskName (char *TaskName, CFE_ES_TaskId_t TaskId, size_t BufferLength)`

Get a Task name for a specified Task ID.
- `CFE_Status_t CFE_ES_DeleteChildTask (CFE_ES_TaskId_t TaskId)`

Deletes a task under an existing Application.
- `void CFE_ES_ExitChildTask (void)`

Exits a child task.

10.17.1 Detailed Description

10.17.2 Function Documentation

10.17.2.1 CFE_ES_CreateChildTask() `CFE_Status_t CFE_ES_CreateChildTask (`

```
    CFE_ES_TaskId_t * TaskIdPtr,
    const char * TaskName,
    CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr,
    CFE_ES_StackPointer_t StackPtr,
    size_t StackSize,
    CFE_ES_TaskPriority_Atom_t Priority,
    uint32 Flags )
```

Creates a new task under an existing Application.

Description

This routine creates a new task (a separate execution thread) owned by the calling Application.

Assumptions, External Events, and Notes:

None

Parameters

<code>out</code>	<code>TaskIdPtr</code>	A pointer to a variable that will be filled in with the new task's ID (must not be null). TaskIdPtr is the Task ID of the newly created child task.
<code>in</code>	<code>TaskName</code>	A pointer to a string containing the desired name of the new task (must not be null). This can be up to <code>OS_MAX_API_NAME</code> characters, including the trailing null.
<code>in</code>	<code>FunctionPtr</code>	A pointer to the function that will be spawned as a new task (must not be null).
<code>in</code>	<code>StackPtr</code>	A pointer to the location where the child task's stack pointer should start. NOTE: Not all underlying operating systems support this parameter. The <code>CFE_ES_TASK_STACK_ALLOCATE</code> constant may be passed to indicate that the stack should be dynamically allocated.
<code>in</code>	<code>StackSize</code>	The number of bytes to allocate for the new task's stack (must not be zero).

Parameters

in	<i>Priority</i>	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority.
in	<i>Flags</i>	Reserved for future expansion.

ReturnsExecution status, see [cFE Return Code Defines](#)**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_CHILD_TASK_CREATE</i>	Child Task Create Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

See also[CFE_ES_DeleteChildTask](#), [CFE_ES_ExitChildTask](#)**10.17.2.2 CFE_ES_DeleteChildTask()** [CFE_Status_t](#) CFE_ES_DeleteChildTask ([CFE_ES_TaskId_t](#) *TaskId*)

Deletes a task under an existing Application.

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	<i>TaskId</i>	The task ID previously obtained when the Child Task was created with the CFE_ES_CreateChildTask API.
----	---------------	--

ReturnsExecution status, see [cFE Return Code Defines](#)**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_CHILD_TASK_DELETE</i>	(return value only verified in coverage test) Child Task Delete Error.
<i>CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK</i>	Child Task Delete Passed Main Task.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

See also

[CFE_ES_CreateChildTask](#), [CFE_ES_ExitChildTask](#)

10.17.2.3 CFE_ES_ExitChildTask() `void CFE_ES_ExitChildTask (`
 `void)`

Exits a child task.

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.

Note

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.

See also

[CFE_ES_CreateChildTask](#), [CFE_ES_DeleteChildTask](#)

10.17.2.4 CFE_ES_GetTaskIDByName() `CFE_Status_t CFE_ES_GetTaskIDByName (`
 `CFE_ES_TaskId_t * TaskIdPtr,`
 `const char * TaskName)`

Get a Task ID associated with a specified Task name.

Description

This routine retrieves the cFE Task ID associated with a specified Task name.

Assumptions, External Events, and Notes:

None

Parameters

<code>out</code>	<code>TaskIdPtr</code>	Pointer to variable that is to receive the Task's ID (must not be null).
<code>in</code>	<code>TaskName</code>	Pointer to null terminated character string containing a Task name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_NAME_NOT_FOUND</code>	Resource Name Error.

Return values

CFE_ES_BAD_ARGUMENT	Bad Argument.
-------------------------------------	---------------

See also

[CFE_ES_GetTaskName](#)

```
10.17.2.5 CFE_ES_GetTaskName() CFE_Status_t CFE_ES_GetTaskName (
    char * TaskName,
    CFE_ES_TaskId_t TaskId,
    size_t BufferLength )
```

Get a Task name for a specified Task ID.

Description

This routine retrieves the cFE Task name associated with a specified Task ID.

Assumptions, External Events, and Notes:

In the case of a failure ([CFE_ES_ERR_RESOURCEID_NOT_VALID](#)), an empty string is returned.

Parameters

out	<i>TaskName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Task name.
in	<i>TaskId</i>	Task ID of Task whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>TaskName</i> buffer. This routine will truncate the name to this length, if necessary.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_GetTaskIDByName](#)

10.18 cFE Miscellaneous APIs

Functions

- void [CFE_ES_BackgroundWakeup](#) (void)
Wakes up the CFE background task.
- [CFE_Status_t CFE_ES_WriteToSysLog](#) (const char *SpecStringPtr,...) [OS_PRINTF](#)(1
Write a string to the cFE System Log.
- [CFE_Status_t uint32 CFE_ES_CalculateCRC](#) (const void *DataPtr, size_t DataLength, uint32 InputCRC,
[CFE_ES_CrcType_Enum_t](#) TypeCRC)
Calculate a CRC on a block of memory.
- void [CFE_ES_ProcessAsyncEvent](#) (void)
Notification that an asynchronous event was detected by the underlying OS/PSP.

10.18.1 Detailed Description

10.18.2 Function Documentation

10.18.2.1 [CFE_ES_BackgroundWakeup\(\)](#) void CFE_ES_BackgroundWakeup (

 void)

Wakes up the CFE background task.

Description

Normally the ES background task wakes up at a periodic interval. Whenever new background work is added, this can be used to wake the task early, which may reduce the delay between adding the job and the job getting processed.

Assumptions, External Events, and Notes:

Note the amount of work that the background task will perform is pro-rated based on the amount of time elapsed since the last wakeup. Waking the task early will not cause the background task to do more work than it otherwise would - it just reduces the delay before work starts initially.

10.18.2.2 [CFE_ES_CalculateCRC\(\)](#) [CFE_Status_t uint32 CFE_ES_CalculateCRC](#) (

 const void * DataPtr,
 size_t DataLength,
 uint32 InputCRC,
 [CFE_ES_CrcType_Enum_t](#) TypeCRC)

Calculate a CRC on a block of memory.

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	<i>DataPtr</i>	Pointer to the base of the memory block.
in	<i>DataLength</i>	The number of bytes in the memory block.
in	<i>InputCRC</i>	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	<i>TypeCRC</i>	One of the following CRC algorithm selections defined in CFE_ES_CrcType_Enum_t

Returns

The result of the CRC calculation on the specified memory block. If the TypeCRC is unimplemented will return 0. If DataPtr is null or DataLength is 0, will return InputCRC

10.18.2.3 CFE_ES_ProcessAsyncEvent() `void CFE_ES_ProcessAsyncEvent (void)`

Notification that an asynchronous event was detected by the underlying OS/PSP.

Description

This hook routine is called from the PSP when an exception or other asynchronous system event occurs

Assumptions, External Events, and Notes:

The PSP must guarantee that this function is only invoked from a context which may use OSAL primitives. In general this means that it shouldn't be *directly* invoked from an ISR/signal context.

10.18.2.4 CFE_ES_WriteToSysLog() `CFE_Status_t CFE_ES_WriteToSysLog (const char * SpecStringPtr, ...)`

Write a string to the cFE System Log.

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	<i>SpecStringPtr</i>	The format string for the log message (must not be null). This is similar to the format string for a printf() call.
----	----------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_SYS_LOG_FULL</i>	System Log Full.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

Referenced by CF_AppInit().

10.19 cFE Critical Data Store APIs

Functions

- [CFE_Status_t CFE_ES_RegisterCDS \(CFE_ES_CDSHandle_t *CDSHandlePtr, size_t BlockSize, const char *Name\)](#)
Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
- [CFE_Status_t CFE_ES_GetCDSBlockIDByName \(CFE_ES_CDSHandle_t *BlockIdPtr, const char *BlockName\)](#)
Get a CDS Block ID associated with a specified CDS Block name.
- [CFE_Status_t CFE_ES_GetCDSBlockName \(char *BlockName, CFE_ES_CDSHandle_t BlockId, size_t BufferLength\)](#)
Get a Block name for a specified Block ID.
- [CFE_Status_t CFE_ES_CopyToCDS \(CFE_ES_CDSHandle_t Handle, const void *DataToCopy\)](#)
Save a block of data in the Critical Data Store (CDS)
- [CFE_Status_t CFE_ES_RestoreFromCDS \(void *RestoreToMemory, CFE_ES_CDSHandle_t Handle\)](#)
Recover a block of data from the Critical Data Store (CDS)

10.19.1 Detailed Description

10.19.2 Function Documentation

10.19.2.1 CFE_ES_CopyToCDS() [CFE_Status_t CFE_ES_CopyToCDS \(](#)
[CFE_ES_CDSHandle_t Handle,](#)
[const void * DataToCopy \)](#)

Save a block of data in the Critical Data Store (CDS)

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	<i>Handle</i>	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS .
in	<i>DataToCopy</i>	A Pointer to the block of memory to be copied into the CDS (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_RegisterCDS](#), [CFE_ES_RestoreFromCDS](#)

10.19.2.2 CFE_ES_GetCDSBlockIDByName() [CFE_Status_t](#) CFE_ES_GetCDSBlockIDByName (

```
    CFE_ES_CDSHandle_t * BlockIdPtr,  
    const char * BlockName )
```

Get a CDS Block ID associated with a specified CDS Block name.

Description

This routine retrieves the CDS Block ID associated with a specified CDS Block name.

Assumptions, External Events, and Notes:

None

Parameters

out	<i>BlockIdPtr</i>	Pointer to variable that is to receive the CDS Block ID (must not be null).
in	<i>BlockName</i>	Pointer to null terminated character string containing a CDS Block name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_NAME_NOT_FOUND	Resource Name Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.

See also

[CFE_ES_GetCDSBlockName](#)

10.19.2.3 CFE_ES_GetCDSBlockName() [CFE_Status_t](#) CFE_ES_GetCDSBlockName (

```
    char * BlockName,  
    CFE_ES_CDSHandle_t BlockId,  
    size_t BufferLength )
```

Get a Block name for a specified Block ID.

Description

This routine retrieves the cFE Block name associated with a specified Block ID.

Assumptions, External Events, and Notes:

In the case of a failure ([CFE_ES_ERR_RESOURCEID_NOT_VALID](#)), an empty string is returned.

Parameters

<i>out</i>	<i>BlockName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the CDS Block name.
<i>in</i>	<i>BlockId</i>	Block ID/Handle of CDS registry entry whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>BlockName</i> buffer. This routine will truncate the name to this length, if necessary.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_NOT_IMPLEMENTED</i>	The processor does not support a Critical Data Store.

See also

[CFE_ES_GetCDSBlockIDByName](#)

10.19.2.4 CFE_ES_RegisterCDS() [*CFE_Status_t*](#) *CFE_ES_RegisterCDS* (

```
CFE_ES_CDSHandle_t * CDSHandlePtr,
size_t BlockSize,
const char * Name )
```

Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)

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:

This function does *not* clear or otherwise initialize/modify the data within the CDS block. If this function returns [*CFE_ES_CDS_ALREADY_EXISTS*](#) the block may already have valid data in it.

If a new CDS block is reserved (either because the name did not exist, or existed as a different size) it is the responsibility of the calling application to fill the CDS block with valid data. This is indicated by a [*CFE_SUCCESS*](#) return code, and in this case the calling application should ensure that it also calls [CFE_ES_CopyToCDS\(\)](#) to fill the block with valid data.

Parameters

<i>out</i>	<i>CDSHandlePtr</i>	Pointer Application's variable that will contain the CDS Memory Block Handle (must not be null). HandlePtr is the handle of the CDS block that can be used in CFE_ES_CopyToCDS and CFE_ES_RestoreFromCDS .
<i>in</i>	<i>BlockSize</i>	The number of bytes needed in the CDS (must not be zero).
<i>in</i>	<i>Name</i>	A pointer to a character string (must not be null) containing an application unique name of <i>CFE_MISSION_ES_CDS_MAX_NAME_LENGTH</i> characters or less.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	The memory block was successfully created in the CDS.
<i>CFE_ES_NOT_IMPLEMENTED</i>	The processor does not support a Critical Data Store.
<i>CFE_ES_CDS_ALREADY_EXISTS</i>	CDS Already Exists.
<i>CFE_ES_CDS_INVALID_SIZE</i>	CDS Invalid Size.
<i>CFE_ES_CDS_INVALID_NAME</i>	CDS Invalid Name.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_CDS_INVALID</i>	(return value only verified in coverage test) CDS Invalid.

See also

[CFE_ES_CopyToCDS](#), [CFE_ES_RestoreFromCDS](#)

10.19.2.5 CFE_ES_RestoreFromCDS() *CFE_Status_t* CFE_ES_RestoreFromCDS (

```
void * RestoreToMemory,
CFE_ES_CDSHandle_t Handle )
```

Recover a block of data from the Critical Data Store (CDS)

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

<i>in</i>	<i>Handle</i>	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS .
<i>out</i>	<i>RestoreToMemory</i>	A Pointer to the block of memory (must not be null) that is to be restored with the contents of the CDS. <i>*RestoreToMemory</i> is the contents of the specified CDS.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

Return values

<i>CFE_ES_CDS_BLOCK_CRC_ERR</i>	(return value only verified in coverage test) CDS Block CRC Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_RegisterCDS](#), [CFE_ES_CopyToCDS](#)

10.20 cFE Memory Manager APIs

Functions

- `CFE_Status_t CFE_ES_PoolCreateNoSem (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`
Initializes a memory pool created by an application without using a semaphore during processing.
- `CFE_Status_t CFE_ES_PoolCreate (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`
Initializes a memory pool created by an application while using a semaphore during processing.
- `CFE_Status_t CFE_ES_PoolCreateEx (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size, uint16 NumBlockSizes, const size_t *BlockSizes, bool UseMutex)`
Initializes a memory pool created by an application with application specified block sizes.
- `int32 CFE_ES_PoolDelete (CFE_ES_MemHandle_t PoolID)`
Deletes a memory pool that was previously created.
- `int32 CFE_ES_GetPoolBuf (CFE_ES_MemPoolBuf_t *BufPtr, CFE_ES_MemHandle_t Handle, size_t Size)`
Gets a buffer from the memory pool created by `CFE_ES_PoolCreate` or `CFE_ES_PoolCreateNoSem`.
- `CFE_Status_t CFE_ES_GetPoolBufInfo (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`
Gets info on a buffer previously allocated via `CFE_ES_GetPoolBuf`.
- `int32 CFE_ES_PutPoolBuf (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`
Releases a buffer from the memory pool that was previously allocated via `CFE_ES_GetPoolBuf`.
- `CFE_Status_t CFE_ES_GetMemPoolStats (CFE_ES_MemPoolStats_t *BufPtr, CFE_ES_MemHandle_t Handle)`
Extracts the statistics maintained by the memory pool software.

10.20.1 Detailed Description

10.20.2 Function Documentation

10.20.2.1 CFE_ES_GetMemPoolStats() `CFE_Status_t CFE_ES_GetMemPoolStats (`
 `CFE_ES_MemPoolStats_t * BufPtr,`
 `CFE_ES_MemHandle_t Handle)`

Extracts the statistics maintained by the memory pool software.

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

<code>out</code>	<code>BufPtr</code>	Pointer to <code>CFE_ES_MemPoolStats_t</code> data structure (must not be null) to be filled with memory statistics. <code>*BufPtr</code> is the Memory Pool Statistics stored in given data structure.
<code>in</code>	<code>Handle</code>	The handle to the memory pool whose statistics are desired.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#)

```
10.20.2.2 CFE_ES_GetPoolBuf() int32 CFE_ES_GetPoolBuf (
    CFE_ES_MemPoolBuf_t * BufPtr,
    CFE_ES_MemHandle_t Handle,
    size_t Size )
```

Gets a buffer from the memory pool created by [CFE_ES_PoolCreate](#) or [CFE_ES_PoolCreateNoSem](#).

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

out	<i>BufPtr</i>	A pointer to the Application's pointer (must not be null) in which will be stored the address of the allocated memory buffer. *BufPtr is the address of the requested buffer.
in	<i>Handle</i>	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem .
in	<i>Size</i>	The size of the buffer requested. NOTE: The size allocated may be larger.

Returns

Bytes Allocated, or error code cFE Return Code Defines

Return values

<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_ERR_MEM_BLOCK_SIZE</i>	Memory Block Size Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#), [CFE_ES_GetPoolBufInfo](#)

```
10.20.2.3 CFE_ES_GetPoolBufInfo() CFE_Status_t CFE_ES_GetPoolBufInfo (
```

```
CFE_ES_MemHandle_t Handle,  
CFE_ES_MemPoolBuf_t BufPtr )
```

Gets info on a buffer previously allocated via [CFE_ES_GetPoolBuf](#).

Description

This routine gets info on a buffer in the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>Handle</i>	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem .
in	<i>BufPtr</i>	A pointer to the memory buffer to provide status for (must not be null).

Returns

Size of the buffer if successful, or status code if not successful, see [cFE Return Code Defines](#)

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BUFFER_NOT_IN_POOL	Buffer Not In Pool.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_GetMemPoolStats](#), [CFE_ES_PutPoolBuf](#)

```
10.20.2.4 CFE_ES_PoolCreate() CFE_Status_t CFE_ES_PoolCreate (  
    CFE_ES_MemHandle_t * PoolID,  
    void * MemPtr,  
    size_t Size )
```

Initializes a memory pool created by an application while using a semaphore during processing.

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

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The CFE_ES_STATIC_POOL_TYPE macro may be used to assist in creating properly aligned memory pools.
<i>in</i>	<i>Size</i>	The size of the pool of memory (must not be zero). Note that this must be an integral multiple of the memory alignment of the processor architecture.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

```
10.20.2.5 CFE_ES_PoolCreateEx() CFE\_Status\_t CFE_ES_PoolCreateEx (
    CFE\_ES\_MemHandle\_t * PoolID,
    void * MemPtr,
    size_t Size,
    uint16 NumBlockSizes,
    const size_t * BlockSizes,
    bool UseMutex )
```

Initializes a memory pool created by an application with application specified block sizes.

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

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The CFE_ES_STATIC_POOL_TYPE macro may be used to assist in creating properly aligned memory pools.

Parameters

in	<i>Size</i>	The size of the pool of memory (must not be zero). Note that this must be an integral multiple of the memory alignment of the processor architecture.
in	<i>NumBlockSizes</i>	The number of different block sizes specified in the <i>BlockSizes</i> array. If set larger than <code>CFE_PLATFORM_ES_POOL_MAX_BUCKETS</code> , <code>CFE_ES_BAD_ARGUMENT</code> will be returned. If <i>BlockSizes</i> is null and <i>NumBlockSizes</i> is 0, <i>NubBlockSizes</i> will be set to <code>CFE_PLATFORM_ES_POOL_MAX_BUCKETS</code> .
in	<i>BlockSizes</i>	Pointer to an array of sizes to be used instead of the default block sizes specified by <code>CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01</code> through <code>CFE_PLATFORM_ES_MAX_BLOCK_SIZE</code> . If the pointer is equal to NULL, the default block sizes are used.
in	<i>UseMutex</i>	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are <code>CFE_ES_USE_MUTEX</code> and <code>CFE_ES_NO_MUTEX</code>

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_ES_NO_RESOURCE_IDS_AVAILABLE</code>	Resource ID is not available.
<code>CFE_STATUS_EXTERNAL_RESOURCE_FAIL</code>	(return value only verified in coverage test) External failure.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

10.20.2.6 CFE_ES_PoolCreateNoSem() `CFE_Status_t CFE_ES_PoolCreateNoSem (`

```
    CFE_ES_MemHandle_t * PoolID,
    void * MemPtr,
    size_t Size )
```

Initializes a memory pool created by an application without using a semaphore during processing.

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

out	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
in	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The CFE_ES_STATIC_POOL_TYPE macro may be used to assist in creating properly aligned memory pools.
in	<i>Size</i>	The size of the pool of memory (must not be zero). Note that this must be an integral multiple of the memory alignment of the processor architecture.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

10.20.2.7 CFE_ES_PoolDelete()

```
int32 CFE_ES_PoolDelete (
    CFE_ES_MemHandle_t PoolID )
```

Deletes a memory pool that was previously created.

Description

This routine removes the pool ID and frees the global table entry for future re-use.

Assumptions, External Events, and Notes:

All buffers associated with the pool become invalid after this call. The application should ensure that buffers/references to the pool are returned before deleting the pool.

Parameters

in	<i>PoolID</i>	The ID of the pool to delete
----	---------------	------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

10.20.2.8 CFE_ES_PutPoolBuf() `int32 CFE_ES_PutPoolBuf (`
 `CFE_ES_MemHandle_t Handle,`
 `CFE_ES_MemPoolBuf_t BufPtr)`

Releases a buffer from the memory pool that was previously allocated via [CFE_ES_GetPoolBuf](#).

Description

This routine releases a buffer back into the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>Handle</i>	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem .
in	<i>BufPtr</i>	A pointer to the memory buffer to be released (must not be null).

Returns

Bytes released, or error code cFE Return Code Defines

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_BUFFER_NOT_IN_POOL	Buffer Not In Pool.
CFE_ES_POOL_BLOCK_INVALID	Invalid pool block.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_GetMemPoolStats](#),
[CFE_ES_GetPoolBufInfo](#)

10.21 cFE Performance Monitor APIs

Macros

- `#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.

Functions

- void `CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)`
Adds a new entry to the data buffer.

10.21.1 Detailed Description

10.21.2 Macro Definition Documentation

10.21.2.1 CFE_ES_PerfLogEntry `#define CFE_ES_PerfLogEntry(` `id) (CFE_ES_PerfLogAdd(id, 0))`

Entry marker for use with Software Performance Analysis Tool.

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.

Assumptions, External Events, and Notes:

None

Parameters

in	<code>id</code>	Identifier of the specific event or marker.
----	-----------------	---

See also

`CFE_ES_PerfLogExit`, `CFE_ES_PerfLogAdd`

Definition at line 1464 of file `cfe_es.h`.

10.21.2.2 CFE_ES_PerfLogExit `#define CFE_ES_PerfLogExit(` `id) (CFE_ES_PerfLogAdd(id, 1))`

Exit marker for use with Software Performance Analysis Tool.

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.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>id</i>	Identifier of the specific event or marker.
----	-----------	---

See also

[CFE_ES_PerfLogEntry](#), [CFE_ES_PerfLogAdd](#)

Definition at line 1483 of file cfe_es.h.

10.21.3 Function Documentation

10.21.3.1 CFE_ES_PerfLogAdd() void CFE_ES_PerfLogAdd (

```
    uint32 Marker,
    uint32 EntryExit )
```

Adds a new entry to the data buffer.

Function called by [CFE_ES_PerfLogEntry](#) and [CFE_ES_PerfLogExit](#) macros

Description

This function logs the entry and exit marker for the specified *id*. This function is used by the Software Performance Analysis tool.

Assumptions, External Events, and Notes:

Marker limited to the range of 0 to [CFE_MISSION_ES_PERF_MAX_IDS](#) - 1. Any performance ids outside of this range will be ignored and will be flagged as an error.

This function implements a circular buffer using an array. DataStart points to first stored entry DataEnd points to next available entry if DataStart == DataEnd then the buffer is either empty or full depending on the value of the DataCount. Time is stored as 2 32 bit integers, (TimerLower32, TimerUpper32): TimerLower32 is the current value of the hardware timer register. TimerUpper32 is the number of times the timer has rolled over.

Parameters

in	<i>Marker</i>	Identifier of the specific event or marker.
in	<i>EntryExit</i>	Used to specify Entry(0) or Exit(1)

See also

[CFE_ES_PerfLogEntry](#), [CFE_ES_PerfLogExit](#)

10.22 cFE Generic Counter APIs

Functions

- `CFE_Status_t CFE_ES_RegisterGenCounter (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`
Register a generic counter.
- `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId)`
Delete a generic counter.
- `CFE_Status_t CFE_ES_IncrementGenCounter (CFE_ES_CounterId_t CounterId)`
Increments the specified generic counter.
- `CFE_Status_t CFE_ES_SetGenCount (CFE_ES_CounterId_t CounterId, uint32 Count)`
Set the specified generic counter.
- `CFE_Status_t CFE_ES_GetGenCount (CFE_ES_CounterId_t CounterId, uint32 *Count)`
Get the specified generic counter count.
- `CFE_Status_t CFE_ES_GetGenCounterIDByName (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`
Get the Id associated with a generic counter name.
- `CFE_Status_t CFE_ES_GetGenCounterName (char *CounterName, CFE_ES_CounterId_t CounterId, size_t BufferLength)`
Get a Counter name for a specified Counter ID.

10.22.1 Detailed Description

10.22.2 Function Documentation

10.22.2.1 CFE_ES_DeleteGenCounter() `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId)`

Delete a generic counter.

Description

This routine deletes a previously registered generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	<code>CounterId</code>	The Counter Id of the newly created counter.
----	------------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

See also

[CFE_ES_IncrementGenCounter](#), [CFE_ES_RegisterGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_GetGenCount](#),
[CFE_ES_GetGenCounterIDByName](#)

10.22.2.2 CFE_ES_GetGenCount() `CFE_Status_t CFE_ES_GetGenCount (`
 `CFE_ES_CounterId_t CounterId,`
 `uint32 * Count)`

Get the specified generic counter count.

Description

This routine gets the value of a generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	<i>CounterId</i>	The Counter to get the value from.
out	<i>Count</i>	Buffer to store value of the Counter (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_RegisterGenCounter](#), [CFE_ES_DeleteGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_IncrementGenCounter](#),
[CFE_ES_GetGenCounterIDByName](#)

10.22.2.3 CFE_ES_GetGenCounterIDByName() `CFE_Status_t CFE_ES_GetGenCounterIDByName (`
 `CFE_ES_CounterId_t * CounterIdPtr,`
 `const char * CounterName)`

Get the Id associated with a generic counter name.

Description

This routine gets the Counter Id for a generic counter specified by name.

Assumptions, External Events, and Notes:

None.

Parameters

<i>out</i>	<i>CounterIdPtr</i>	Pointer to variable that is to receive the Counter's ID (must not be null).
<i>in</i>	<i>CounterName</i>	Pointer to null terminated character string containing a Counter name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_NAME_NOT_FOUND</i>	Resource Name Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_GetGenCounterName](#)

10.22.2.4 CFE_ES_GetGenCounterName() *CFE_Status_t* CFE_ES_GetGenCounterName (

```
    char * CounterName,
    CFE_ES_CounterId_t CounterId,
    size_t BufferLength )
```

Get a Counter name for a specified Counter ID.

Description

This routine retrieves the cFE Counter name associated with a specified Counter ID.

Assumptions, External Events, and Notes:

In the case of a failure ([CFE_ES_ERR_RESOURCEID_NOT_VALID](#)), an empty string is returned.

Parameters

<i>out</i>	<i>CounterName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Counter name.
<i>in</i>	<i>CounterId</i>	ID of Counter whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator (must not be zero), that can be put into the <i>CounterName</i> buffer. This routine will truncate the name to this length, if necessary.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
--------------------	-----------------------

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also[CFE_ES_GetGenCounterIDByName](#)**10.22.2.5 CFE_ES_IncrementGenCounter()** [CFE_Status_t](#) CFE_ES_IncrementGenCounter ([CFE_ES_CounterId_t](#) CounterId)

Increments the specified generic counter.

Description

This routine increments the specified generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	<i>CounterId</i>	The Counter to be incremented.
--------------------	------------------	--------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also[CFE_ES_RegisterGenCounter](#), [CFE_ES_DeleteGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_GetGenCount](#), [CFE_ES_GetGenCounterIDByName](#)**10.22.2.6 CFE_ES_RegisterGenCounter()** [CFE_Status_t](#) CFE_ES_RegisterGenCounter ([CFE_ES_CounterId_t](#) * CounterIdPtr, [const char](#) * CounterName)

Register a generic counter.

Description

This routine registers a generic thread-safe counter which can be used for inter-task management.

Assumptions, External Events, and Notes:

The initial value of all newly registered counters is 0.

Parameters

out	<i>CounterIdPtr</i>	Buffer to store the Counter Id of the newly created counter (must not be null).
in	<i>CounterName</i>	The Name of the generic counter (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_ERR_DUPLICATE_NAME</i>	Duplicate Name Error.
<i>CFE_ES_NO_RESOURCE_IDS_AVAILABLE</i>	Resource ID is not available.

See also

[CFE_ES_IncrementGenCounter](#), [CFE_ES_DeleteGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_GetGenCount](#), [CFE_ES_GetGenCounterIDByName](#)

10.22.2.7 CFE_ES_SetGenCount() *CFE_Status_t* CFE_ES_SetGenCount (
CFE_ES_CounterId_t *CounterId*,
uint32 *Count*)

Set the specified generic counter.

Description

This routine sets the specified generic counter to the specified value.

Assumptions, External Events, and Notes:

None.

Parameters

in	<i>CounterId</i>	The Counter to be set.
in	<i>Count</i>	The new value of the Counter.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_RegisterGenCounter](#), [CFE_ES_DeleteGenCounter](#), [CFE_ES_IncrementGenCounter](#), [CFE_ES_GetGenCount](#),
[CFE_ES_GetGenCounterIDByName](#)

10.23 cFE Registration APIs

Functions

- **CFE_Status_t CFE_EVS_Register** (const void *Filters, uint16 NumEventFilters, uint16 FilterScheme)
Register an application for receiving event services.

10.23.1 Detailed Description

10.23.2 Function Documentation

10.23.2.1 CFE_EVS_Register() `CFE_Status_t CFE_EVS_Register (`

```
    const void * Filters,
    uint16 NumEventFilters,
    uint16 FilterScheme )
```

Register an application for receiving event services.

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 CFE_EVS_BinFilter {
    uint16 EventID,
    uint16 Mask ;
} CFE_EVS_BinFilter_t;
```

Parameters

in	<i>Filters</i>	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	<i>NumEventFilters</i>	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	<i>FilterScheme</i>	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.

Returns

Execution status below or from **CFE_ES_GetAppID**, see **cFE Return Code Defines**

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_EVS_APP_FILTER_OVERLOAD</i>	Application Filter Overload.
<i>CFE_EVS_UNKNOWN_FILTER</i>	Unknown Filter.
<i>CFE_EVS_APP_ILLEGAL_APP_ID</i>	Illegal Application ID.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

Referenced by CF_AppInit().

10.24 cFE Send Event APIs

Functions

- `CFE_Status_t CFE_EVS_SendEvent (uint16 EventID, CFE_EVS_EventType_Enum_t EventType, const char *Spec,...) OS_PRINTF(3)`
Generate a software event.
- `CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (uint16 EventID, CFE_EVS_EventType_Enum_t EventType, CFE_ES_AppId_t AppID, const char *Spec,...) OS_PRINTF(4)`
Generate a software event given the specified Application ID.
- `CFE_Status_t CFE_Status_t CFE_Status_t CFE_EVS_SendTimedEvent (CFE_TIME_SysTime_t Time, uint16 EventID, CFE_EVS_EventType_Enum_t EventType, const char *Spec,...) OS_PRINTF(4)`
Generate a software event with a specific time tag.

10.24.1 Detailed Description

10.24.2 Function Documentation

10.24.2.1 CFE_EVS_SendEvent() `CFE_Status_t CFE_EVS_SendEvent (`
`uint16 EventID,`
`CFE_EVS_EventType_Enum_t EventType,`
`const char * Spec,`
`...)`

Generate a software event.

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:

This API only works within the context of a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) `CFE_ES_WriteToSysLog` can be used for reporting.

Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <code>EventID</code> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"> • <code>CFE_EVS_EventType_DEBUG</code> • <code>CFE_EVS_EventType_INFORMATION</code> • <code>CFE_EVS_EventType_ERROR</code> • <code>CFE_EVS_EventType_CRITICAL</code>

Parameters

in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <code>CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</code> . 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.
----	-------------	--

Returns

Execution status, see [CFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_EVS_APP_NOT_REGISTERED</code>	Application Not Registered.
<code>CFE_EVS_APP_ILLEGAL_APP_ID</code>	Illegal Application ID.
<code>CFE_EVS_INVALID_PARAMETER</code>	Invalid Pointer.

See also

[CFE_EVS_SendEventWithAppID](#), [CFE_EVS_SendTimedEvent](#)

Referenced by `CF_AbandonCmd()`, `CF_AppInit()`, `CF_AppMain()`, `CF_AppPipe()`, `CF_CancelCmd()`, `CF_CFDP_InitEngine()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_PlaybackDir()`, `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_R2_CalcCrcChunk()`, `CF_CFDP_R2_Complete()`, `CF_CFDP_R2_Recv_fin_ack()`, `CF_CFDP_R2_RecvMd()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_R_Init()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendInactivityEvent()`, `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_R_SubstateSendNak()`, `CF_CFDP_R_Tick()`, `CF_CFDP_ReceiveMessage()`, `CF_CFDP_RecvAck()`, `CF_CFDP_RecvEof()`, `CF_CFDP_RecvFd()`, `CF_CFDP_RecvFin()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_RecvMd()`, `CF_CFDP_RecvNak()`, `CF_CFDP_RecvPh()`, `CF_CFD_P_ResetTransaction()`, `CF_CFDP_S2_EarlyFin()`, `CF_CFDP_S2_Nak()`, `CF_CFDP_S2_WaitForEofAck()`, `CF_CFDP_S_DispatchRecv()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_S_SubstateSendMetadata()`, `CF_CFDP_S_Tick()`, `CF_CFDP_TxFile()`, `CF_CFDP_TxFile_Initiate()`, `CF_CheckTables()`, `CF_DisableDequeueCmd()`, `CF_DisableDirPollingCmd()`, `CF_DisableEngineCmd()`, `CF_DoChanAction()`, `CF_DoEnableDisablePolldir()`, `CF_DoPurgeQueue()`, `CF_DoSuspRes()`, `CF_EnableDequeueCmd()`, `CF_EnableDirPollingCmd()`, `CF_EnableEngineCmd()`, `CF_FreezeCmd()`, `CF_GetSetParamCmd()`, `CF_NoopCmd()`, `CF_PlaybackDirCmd()`, `CF_ProcessGroundCommand()`, `CF_PurgeQueueCmd()`, `CF_ResetCountersCmd()`, `CF_TableInit()`, `CF_ThawCmd()`, `CF_TsnChanAction()`, `CF_TxFileCmd()`, `CF_ValidateConfigTable()`, `CF_WrappedClose()`, `CF_WriteHistoryEntryToFile()`, and `CF_WriteQueueCmd()`.

```
10.24.2.2 CFE_EVS_SendEventWithAppID() CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (
    uint16 EventID,
    CFE_EVS_EventType_Enum_t EventType,
    CFE_ES_AppId_t AppID,
    const char * Spec,
    ... )
```

Generate a software event given the specified Application ID.

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:

The Application ID must correspond to a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE_ES_WriteToSysLog](#) can be used for reporting.

Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The EventID is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"> • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	<i>AppID</i>	The Application ID from which the event message should appear.
in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> 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.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_NOT_REGISTERED	Application Not Registered.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_EVS_INVALID_PARAMETER	Invalid Pointer.

See also

[CFE_EVS_SendEvent](#), [CFE_EVS_SendTimedEvent](#)

10.24.2.3 CFE_EVS_SendTimedEvent() `CFE_Status_t CFE_Status_t CFE_Status_t CFE_EVS_SendTimedEvent(Event (`

```
CFE_TIME_SysTime_t Time,
uint16 EventID,
CFE_EVS_EventType_Enum_t EventType,
const char * Spec,
... )
```

Generate a software event with a specific time tag.

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:

This API only works within the context of a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE_ES_WriteToSysLog](#) can be used for reporting.

Parameters

in	<i>Time</i>	The time to include in the event. This will usually be a time returned by the function CFE_TIME_GetTime .
in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <i>EventID</i> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"> • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> 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.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_NOT_REGISTERED	Application Not Registered.

Return values

<i>CFE_EVS_APP_ILLEGAL_APP_ID</i>	Illegal Application ID.
<i>CFE_EVS_INVALID_PARAMETER</i>	Invalid Pointer.

See also

[CFE_EVS_SendEvent](#), [CFE_EVS_SendEventWithAppID](#)

10.25 cFE Reset Event Filter APIs

Functions

- [CFE_Status_t CFE_EVS_ResetFilter \(uint16 EventID\)](#)
Resets the calling application's event filter for a single event ID.
- [CFE_Status_t CFE_EVS_ResetAllFilters \(void\)](#)
Resets all of the calling application's event filters.

10.25.1 Detailed Description

10.25.2 Function Documentation

10.25.2.1 CFE_EVS_ResetAllFilters() [CFE_Status_t CFE_EVS_ResetAllFilters \(void \)](#)

Resets all of the calling application's event filters.

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

Returns

Execution status below or from [CFE_ES_GetAppID](#), see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_NOT_REGISTERED	Application Not Registered.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.

See also

[CFE_EVS_ResetFilter](#)

10.25.2.2 CFE_EVS_ResetFilter() [CFE_Status_t CFE_EVS_ResetFilter \(uint16 EventID \)](#)

Resets the calling application's event filter for a single event ID.

Description

Resets the filter such that the next event is treated like the first. For example, if the filter was set to only send the first event, the next event following the reset would be sent.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The Event ID is defined and supplied by the application sending the event.
----	----------------	--

Returns

Execution status below or from [CFE_ES_GetAppID](#), see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_NOT_REGISTERED	Application Not Registered.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_EVS_EVT_NOT_REGISTERED	Event Not Registered.

See also

[CFE_EVS_ResetAllFilters](#)

10.26 cFE File Header Management APIs

Functions

- `CFE_Status_t CFE_FS_ReadHeader (CFE_FS_Header_t *Hdr, osal_id_t 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.
- `CFE_Status_t CFE_FS_WriteHeader (osal_id_t FileDes, CFE_FS_Header_t *Hdr)`
Write the specified Standard cFE File Header to the specified file.
- `CFE_Status_t CFE_FS_SetTimestamp (osal_id_t FileDes, CFE_TIME_SysTime_t NewTimestamp)`
Modifies the Time Stamp field in the Standard cFE File Header for the specified file.

10.26.1 Detailed Description

10.26.2 Function Documentation

10.26.2.1 CFE_FS_InitHeader() `void CFE_FS_InitHeader (`
 `CFE_FS_Header_t * Hdr,`
 `const char * Description,`
 `uint32 SubType)`

Initializes the contents of the Standard cFE File Header.

Description

This API will clear the specified `CFE_FS_Header_t` variable and initialize the description field with the specified value

Parameters

in	<code>Hdr</code>	Pointer to a variable of type <code>CFE_FS_Header_t</code> that will be cleared and initialized
in	<code>Description</code>	Initializes Header's Description (must not be null)
in	<code>SubType</code>	Initializes Header's SubType

See also

[CFE_FS_WriteHeader](#)

10.26.2.2 CFE_FS_ReadHeader() `CFE_Status_t CFE_FS_ReadHeader (`
 `CFE_FS_Header_t * Hdr,`
 `osal_id_t FileDes)`

Read the contents of the Standard cFE File Header.

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_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. File offset behavior: Agnostic on entry since it will move the offset to the start of the file, on success the offset will be at the end of the header, undefined offset behavior for error cases.

Parameters

<i>out</i>	<i>Hdr</i>	Pointer to a variable of type CFE_FS_Header_t (must not be null) that will be filled with the contents of the Standard cFE File Header. *Hdr is the contents of the Standard cFE File Header for the specified file.
<i>in</i>	<i>FileDes</i>	File Descriptor obtained from a previous call to OS_OpenCreate that is associated with the file whose header is to be read.

Returns

Bytes read or error status from OSAL

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.
-------------------------------------	---------------

Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

See also

[CFE_FS_WriteHeader](#)

10.26.2.3 CFE_FS_SetTimestamp() [CFE_Status_t](#) CFE_FS_SetTimestamp (
osal_id_t *FileDes*,
CFE_TIME_SysTime_t *NewTimestamp*)

Modifies the Time Stamp field in the Standard cFE File Header for the specified file.

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_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. The [NewTimestamp](#) field has been filled appropriately by the Application.
3. File offset behavior: Agnostic on entry since it will move the offset, on success the offset will be at the end of the time stamp, undefined offset behavior for error cases.

Parameters

in	<i>FileDes</i>	File Descriptor obtained from a previous call to OS_OpenCreate that is associated with the file whose header is to be read.
in	<i>NewTimestamp</i>	A CFE_TIME_SysTime_t data structure containing the desired time to be put into the file's Standard cFE File Header.

Returns

Execution status, see [cFE Return Code Defines](#), or OSAL status

Return values

CFE_STATUS_EXTERNAL_RESOURCE_FAIL	(return value only verified in coverage test) External failure.
CFE_SUCCESS	Successful execution.

Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

10.26.2.4 CFE_FS_WriteHeader() [CFE_Status_t](#) CFE_FS_WriteHeader (
 [osal_id_t](#) *FileDes*,
 [CFE_FS_Header_t](#) * *Hdr*)

Write the specified Standard cFE File Header to the specified file.

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_OpenCreate](#) 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.
4. File offset behavior: Agnostic on entry since it will move the offset to the start of the file, on success the offset will be at the end of the header, undefined offset behavior for error cases.

Parameters

in	<i>FileDes</i>	File Descriptor obtained from a previous call to OS_OpenCreate that is associated with the file whose header is to be read.
out	<i>Hdr</i>	Pointer to a variable of type CFE_FS_Header_t (must not be null) that will be filled with the contents of the Standard cFE File Header. *Hdr is the contents of the Standard cFE File Header for the specified file.

Returns

Bytes read or error status from OSAL

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.
-------------------------------------	---------------

Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

See also

[CFE_FS_ReadHeader](#)

10.27 cFE File Utility APIs

Functions

- const char * [CFE_FS_GetDefaultMountPoint](#) (CFE_FS_FileCategory_t FileCategory)
Get the default virtual mount point for a file category.
- const char * [CFE_FS_GetDefaultExtension](#) (CFE_FS_FileCategory_t FileCategory)
Get the default filename extension for a file category.
- int32 [CFE_FS_ParseInputFileNameEx](#) (char *OutputBuffer, const char *InputBuffer, size_t OutputBufSize, size_t InputBufSize, const char *DefaultInput, const char *DefaultPath, const char *DefaultExtension)
Parse a filename input from an input buffer into a local buffer.
- int32 [CFE_FS_ParseInputFileName](#) (char *OutputBuffer, const char *InputName, size_t OutputBufSize, CFE_FS_FileCategory_t FileCategory)
Parse a filename string from the user into a local buffer.
- CFE_Status_t [CFE_FS_ExtractFilenameFromPath](#) (const char *OriginalPath, char *FileNameOnly)
Extracts the filename from a unix style path and filename string.
- int32 [CFE_FS_BackgroundFileDumpRequest](#) (CFE_FS_FileWriteMetaData_t *Meta)
Register a background file dump request.
- bool [CFE_FS_BackgroundFileDumplsPending](#) (const CFE_FS_FileWriteMetaData_t *Meta)
Query if a background file write request is currently pending.

10.27.1 Detailed Description

10.27.2 Function Documentation

10.27.2.1 [CFE_FS_BackgroundFileDumplsPending\(\)](#) bool CFE_FS_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t * Meta)

Query if a background file write request is currently pending.

Description

This returns "true" while the request is on the background work queue. This returns "false" once the request is complete and removed from the queue.

Assumptions, External Events, and Notes:

None

Parameters

in, out	Meta	The background file write persistent state object (must not be null)
---------	------	--

Returns

boolean value indicating if request is already pending

Return values

true	if request is pending
false	if request is not pending

10.27.2.2 CFE_FS_BackgroundFileDumpRequest() `int32 CFE_FS_BackgroundFileDumpRequest (CFE_FS_FileWriteMetaData_t * Meta)`

Register a background file dump request.

Description

Puts the previously-initialized metadata into the pending request queue

Assumptions, External Events, and Notes:

Metadata structure should be stored in a persistent memory area (not on stack) as it must remain accessible by the file writer task throughout the asynchronous job operation.

Parameters

<code>in, out</code>	<code>Meta</code>	The background file write persistent state object (must not be null)
----------------------	-------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_FS_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_FS_INVALID_PATH</code>	Invalid Path.
<code>CFE_STATUS_REQUEST_ALREADY_PENDING</code>	Request already pending.
<code>CFE_SUCCESS</code>	Successful execution.

10.27.2.3 CFE_FS_ExtractFilenameFromPath() `CFE_Status_t CFE_FS_ExtractFilenameFromPath (`

```
const char * OriginalPath,
char * FileNameOnly )
```

Extracts the filename from a unix style path and filename string.

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 (including terminator) is no longer than `OS_MAX_PATH_LEN`

Parameters

<code>in</code>	<code>OriginalPath</code>	The original path (must not be null)
<code>out</code>	<code>FileNameOnly</code>	The filename that is extracted from the path (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.
CFE_FS_FNAME_TOO_LONG	Filename Too Long.
CFE_FS_INVALID_PATH	Invalid Path.
CFE_SUCCESS	Successful execution.

10.27.2.4 CFE_FS_GetDefaultExtension() `const char* CFE_FS_GetDefaultExtension (`
`CFE_FS_FileCategory_t FileCategory)`

Get the default filename extension for a file category.

Certain file types may have an extension that varies from system to system. This is primarily an issue for application modules which are ".so" on Linux systems, ".dll" on Windows, ".o" on VxWorks, ".obj" on RTEMS, and so on.

This uses a combination of compile-time configuration and hints from the build environment to get the default/expected extension for a given file category.

Returns

String containing the extension

Return values

<code>NULL</code>	if no default extension is known for the given file category
-------------------	--

10.27.2.5 CFE_FS_GetDefaultMountPoint() `const char* CFE_FS_GetDefaultMountPoint (`
`CFE_FS_FileCategory_t FileCategory)`

Get the default virtual mount point for a file category.

Certain classes of files generally reside in a common directory, mainly either the persistent storage (/cf typically) or ram disk (/ram typically).

Ephemeral status files are generally in the ram disk while application modules and scripts are generally in the persistent storage.

This returns the expected directory for a given class of files in the form of a virtual OSAL mount point string.

Returns

String containing the mount point

Return values

<code>NULL</code>	if no mount point is known for the given file category
-------------------	--

10.27.2.6 CFE_FS_ParseInputFileName() `int32 CFE_FS_ParseInputFileName (`
`char * OutputBuffer,`

```
const char * InputName,
size_t OutputBufSize,
CFE_FS_FileCategory_t FileCategory)
```

Parse a filename string from the user into a local buffer.

Description

Simplified API for [CFE_FS_ParseInputFileNameEx\(\)](#) where input is always known to be a non-empty, null terminated string and the fixed-length input buffer not needed. For instance this may be used where the input is a fixed string from cfe_platform_cfg.h or similar.

Assumptions, External Events, and Notes:

The parameters are organized such that this is basically like strncpy() with an extra argument, and existing file name accesses which use a direct copy can easily change to use this instead.

See also

[CFE_FS_ParseInputFileNameEx\(\)](#)

Parameters

out	<i>OutputBuffer</i>	Buffer to store result (must not be null).
in	<i>InputName</i>	A null terminated input string (must not be null).
in	<i>OutputBufSize</i>	Maximum Size of output buffer (must not be zero).
in	<i>FileCategory</i>	The generalized category of file (implies default path/extension)

Returns

Execution status, see [cFE Return Code Defines](#)

```
10.27.2.7 CFE_FS_ParseInputFileNameEx() int32 CFE_FS_ParseInputFileNameEx (
    char * OutputBuffer,
    const char * InputBuffer,
    size_t OutputBufSize,
    size_t InputBufSize,
    const char * DefaultInput,
    const char * DefaultPath,
    const char * DefaultExtension)
```

Parse a filename input from an input buffer into a local buffer.

Description

This provides a more user friendly way to specify file names, using default values for the path and extension, which can vary from system to system.

If InputBuffer is null or its length is zero, then DefaultInput is used as if it was the content of the input buffer.
If either the pathname or extension is missing from the input, it will be added from defaults, with the complete fully-qualified filename stored in the output buffer.

Assumptions, External Events, and Notes:

1. The paths and filenames used here are the standard unix style filenames separated by "/" (path) and "." (extension) characters.
2. Input Buffer has a fixed max length. Parsing will not exceed InputBufSize, and does not need to be null terminated. However parsing will stop at the first null char, when the input is shorter than the maximum.

Parameters

out	<i>OutputBuffer</i>	Buffer to store result (must not be null).
in	<i>InputBuffer</i>	A input buffer that may contain a file name (e.g. from command) (must not be null).
in	<i>OutputBufSize</i>	Maximum Size of output buffer (must not be zero).
in	<i>InputBufSize</i>	Maximum Size of input buffer.
in	<i>DefaultInput</i>	Default value to use for input if InputBffer is empty
in	<i>DefaultPath</i>	Default value to use for pathname if omitted from input
in	<i>DefaultExtension</i>	Default value to use for extension if omitted from input

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.
CFE_FS_FNAME_TOO_LONG	Filename Too Long.
CFE_FS_INVALID_PATH	Invalid Path.
CFE_SUCCESS	Successful execution.

10.28 cFE Generic Message APIs

Functions

- `CFE_Status_t CFE_MSG_Init (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId, CFE_MSG_Size_t Size)`

Initialize a message.

10.28.1 Detailed Description

10.28.2 Function Documentation

10.28.2.1 CFE_MSG_Init() `CFE_Status_t CFE_MSG_Init (`

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_SB_MsgId_t MsgId,  
    CFE_MSG_Size_t Size )
```

Initialize a message.

Description

This routine initialize a message. The entire message is set to zero (based on size), defaults are set, then the size and bits from MsgId are set.

Parameters

out	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
in	<code>MsgId</code>	MsgId that corresponds to message
in	<code>Size</code>	Total size of the message (used to set length field)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.

Referenced by CF_AppInit(), CF_CFDP_MsgOutGet(), and CF_CFDP_SendEotPkt().

10.29 cFE Message Primary Header APIs

Functions

- `CFE_Status_t CFE_MSG_GetSize (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t *Size)`
Gets the total size of a message.
- `CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)`
Sets the total size of a message.
- `CFE_Status_t CFE_MSG.GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)`
Gets the message type.
- `CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)`
Sets the message type.
- `CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)`
Gets the message header version.
- `CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)`
Sets the message header version.
- `CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)`
Gets the message secondary header boolean.
- `CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)`
Sets the message secondary header boolean.
- `CFE_Status_t CFE_MSG_GetApld (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t *Apld)`
Gets the message application ID.
- `CFE_Status_t CFE_MSG_SetApld (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t Apld)`
Sets the message application ID.
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`
Gets the message segmentation flag.
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`
Sets the message segmentation flag.
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`
Gets the message sequence count.
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`
Sets the message sequence count.
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`
Gets the next sequence count value (rolls over if appropriate)

10.29.1 Detailed Description

10.29.2 Function Documentation

```
10.29.2.1 CFE_MSG_GetApId() CFE_Status_t CFE_MSG_GetApId (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_ApId_t * ApId )
```

Gets the message application ID.

Description

This routine gets the message application ID.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>ApId</i>	Application ID (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

```
10.29.2.2 CFE_MSG_GetHasSecondaryHeader() CFE_Status_t CFE_MSG_GetHasSecondaryHeader (
    const CFE_MSG_Message_t * MsgPtr,
    bool * HasSecondary )
```

Gets the message secondary header boolean.

Description

This routine gets the message secondary header boolean.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>HasSecondary</i>	Has secondary header flag (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.3 CFE_MSG_GetHeaderVersion() `CFE_Status_t CFE_MSG_GetHeaderVersion (`
 `const CFE_MSG_Message_t * MsgPtr,`
 `CFE_MSG_HeaderVersion_t * Version)`

Gets the message header version.

Description

This routine gets the message header version.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Version</i>	Header version (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.

10.29.2.4 CFE_MSG_GetNextSequenceCount() `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (`
 `CFE_MSG_SequenceCount_t SeqCnt)`

Gets the next sequence count value (rolls over if appropriate)

Description

Abstract method to get the next valid sequence count value. Will roll over to zero for any input value greater than or equal to the maximum possible sequence count value given the field in the header.

Parameters

in	<i>SeqCnt</i>	Sequence count
----	---------------	----------------

Returns

The next valid sequence count value

10.29.2.5 CFE_MSG_GetSegmentationFlag() `CFE_Status_t CFE_MSG_GetSegmentationFlag (`
 `const CFE_MSG_Message_t * MsgPtr,`
 `CFE_MSG_SegmentationFlag_t * SegFlag)`

Gets the message segmentation flag.

Description

This routine gets the message segmentation flag

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>SegFlag</i>	Segmentation flag (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.6 CFE_MSG_GetSequenceCount() *CFE_Status_t* CFE_MSG_GetSequenceCount (

```
const CFE_MSG_Message_t * MsgPtr,
CFE_MSG_SequenceCount_t * SeqCnt )
```

Gets the message sequence count.

Description

This routine gets the message sequence count.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>SeqCnt</i>	Sequence count (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.7 CFE_MSG.GetSize() *CFE_Status_t* CFE_MSG_GetSize (

```
const CFE_MSG_Message_t * MsgPtr,
CFE_MSG_Size_t * Size )
```

Gets the total size of a message.

Description

This routine gets the total size of the message.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Size</i>	Total message size (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

Referenced by CF_AppPipe(), CF_CFDP_ReceiveMessage(), and CF_ProcessGroundCommand().

10.29.2.8 CFE_MSG_GetType() *CFE_Status_t* CFE_MSG_GetType (

```
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_Type_t * Type )
```

Gets the message type.

Description

This routine gets the message type.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Type</i>	Message type (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

Referenced by CF_CFDP_ReceiveMessage().

10.29.2.9 CFE_MSG_SetApId() *CFE_Status_t* CFE_MSG_SetApId (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_ApId_t ApId )
```

Sets the message application ID.

Description

This routine sets the message application ID. Typically set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>ApId</i>	Application ID

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.10 CFE_MSG_SetHasSecondaryHeader() *CFE_Status_t* CFE_MSG_SetHasSecondaryHeader (
 CFE_MSG_Message_t * *MsgPtr*,
 bool *HasSecondary*)

Sets the message secondary header boolean.

Description

This routine sets the message secondary header boolean. Typically only set within message initialization and not used by APPs.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>HasSecondary</i>	Has secondary header flag

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.11 CFE_MSG_SetHeaderVersion() *CFE_Status_t* CFE_MSG_SetHeaderVersion (
 CFE_MSG_Message_t * *MsgPtr*,
 CFE_MSG_HeaderVersion_t *Version*)

Sets the message header version.

Description

This routine sets the message header version. Typically only set within message initialization and not used by APPs.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message.
in	<i>Version</i>	Header version

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.12 CFE_MSG_SetSegmentationFlag() *CFE_Status_t* CFE_MSG_SetSegmentationFlag (
 CFE_MSG_Message_t * *MsgPtr*,
 CFE_MSG_SegmentationFlag_t *SegFlag*)

Sets the message segmentation flag.

Description

This routine sets the message segmentation flag.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>SegFlag</i>	Segmentation flag

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.13 CFE_MSG_SetSequenceCount() *CFE_Status_t* CFE_MSG_SetSequenceCount (
 CFE_MSG_Message_t * *MsgPtr*,

`CFE_MSG_SequenceCount_t SeqCnt)`

Sets the message sequence count.

Description

This routine sets the message sequence count.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>SeqCnt</i>	Sequence count

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.

10.29.2.14 CFE_MSG_SetSize() `CFE_Status_t CFE_MSG_SetSize (`
`CFE_MSG_Message_t * MsgPtr,`
`CFE_MSG_Size_t Size)`

Sets the total size of a message.

Description

This routine sets the total size of the message.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>Size</i>	Total message size

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.

Referenced by CF_CFDP_Send().

10.29.2.15 CFE_MSG_SetType() `CFE_Status_t CFE_MSG_SetType (`

```
CFE_MSG_Message_t * MsgPtr,  
CFE_MSG_Type_t Type )
```

Sets the message type.

Description

This routine sets the message type.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>Type</i>	Message type

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30 cFE Message Extended Header APIs

Functions

- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`
Gets the message EDS version.
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`
Sets the message EDS version.
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`
Gets the message endian.
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`
Sets the message endian.
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`
Gets the message playback flag.
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`
Sets the message playback flag.
- `CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)`
Gets the message subsystem.
- `CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)`
Sets the message subsystem.
- `CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)`
Gets the message system.
- `CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)`
Sets the message system.

10.30.1 Detailed Description

10.30.2 Function Documentation

10.30.2.1 CFE_MSG_GetEDSVersion() `CFE_Status_t CFE_MSG_GetEDSVersion (`
`const CFE_MSG_Message_t * MsgPtr,`
`CFE_MSG_EDSVersion_t * Version)`

Gets the message EDS version.

Description

This routine gets the message EDS version.

Parameters

in	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
out	<code>Version</code>	EDS Version (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.2 CFE_MSG_GetEndian() `CFE_Status_t CFE_MSG_GetEndian (`
 `const CFE_MSG_Message_t * MsgPtr,`
 `CFE_MSG_Endian_t * Endian)`

Gets the message endian.

Description

This routine gets the message endian.

Parameters

<code>in</code>	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
<code>out</code>	<code>Endian</code>	Endian (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.3 CFE_MSG_GetPlaybackFlag() `CFE_Status_t CFE_MSG_GetPlaybackFlag (`
 `const CFE_MSG_Message_t * MsgPtr,`
 `CFE_MSG_PlaybackFlag_t * PlayFlag)`

Gets the message playback flag.

Description

This routine gets the message playback flag.

Parameters

<code>in</code>	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
<code>out</code>	<code>PlayFlag</code>	Playback Flag (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.4 CFE_MSG_GetSubsystem() [CFE_Status_t](#) CFE_MSG_GetSubsystem (

```
const CFE_MSG_Message_t * MsgPtr,  
CFE_MSG_Subsystem_t * Subsystem )
```

Gets the message subsystem.

Description

This routine gets the message subsystem

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Subsystem</i>	Subsystem (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.5 CFE_MSG_GetSystem() [CFE_Status_t](#) CFE_MSG_GetSystem (

```
const CFE_MSG_Message_t * MsgPtr,  
CFE_MSG_System_t * System )
```

Gets the message system.

Description

This routine gets the message system id

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>System</i>	System (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.6 CFE_MSG_SetEDSVersion() [CFE_Status_t](#) CFE_MSG_SetEDSVersion (
 [CFE_MSG_Message_t](#) * *MsgPtr*,
 [CFE_MSG_EDSVersion_t](#) *Version*)

Sets the message EDS version.

Description

This routine sets the message EDS version.

Parameters

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>Version</i>	EDS Version

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.7 CFE_MSG_SetEndian() [CFE_Status_t](#) CFE_MSG_SetEndian (
 [CFE_MSG_Message_t](#) * *MsgPtr*,
 [CFE_MSG_Endian_t](#) *Endian*)

Sets the message endian.

Description

This routine sets the message endian. Invalid endian selection will set big endian.

Parameters

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>Endian</i>	Endian

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.8 CFE_MSG_SetPlaybackFlag() [CFE_Status_t](#) CFE_MSG_SetPlaybackFlag (

[CFE_MSG_Message_t](#) * *MsgPtr*,
[CFE_MSG_PlaybackFlag_t](#) *PlayFlag*)

Sets the message playback flag.

Description

This routine sets the message playback flag.

Parameters

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>PlayFlag</i>	Playback Flag

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.9 CFE_MSG_SetSubsystem() [CFE_Status_t](#) CFE_MSG_SetSubsystem (

[CFE_MSG_Message_t](#) * *MsgPtr*,
[CFE_MSG_Subsystem_t](#) *Subsystem*)

Sets the message subsystem.

Description

This routine sets the message subsystem. Some bits may be set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

Parameters

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>Subsystem</i>	Subsystem

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.30.2.10 CFE_MSG_SetSystem() [`CFE_Status_t CFE_MSG_SetSystem \(`](#)
[`CFE_MSG_Message_t * MsgPtr,`](#)
[`CFE_MSG_System_t System \)`](#)

Sets the message system.

Description

This routine sets the message system id. Some bits may be set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

Parameters

<code>in, out</code>	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
<code>in</code>	<code>System</code>	System

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.31 cFE Message Secondary Header APIs

Functions

- `CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t *MsgPtr)`
Calculates and sets the checksum of a message.
- `CFE_Status_t CFE_MSG_ValidateChecksum (const CFE_MSG_Message_t *MsgPtr, bool *isValid)`
Validates the checksum of a message.
- `CFE_Status_t CFE_MSG_SetFcnCode (CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode)`
Sets the function code field in a message.
- `CFE_Status_t CFE_MSG_GetFcnCode (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *FcnCode)`
Gets the function code field from a message.
- `CFE_Status_t CFE_MSG_GetMsgTime (const CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t *Time)`
Gets the time field from a message.
- `CFE_Status_t CFE_MSG_SetMsgTime (CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t NewTime)`
Sets the time field in a message.

10.31.1 Detailed Description

10.31.2 Function Documentation

10.31.2.1 CFE_MSG_GenerateChecksum() `CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t * MsgPtr)`

Calculates and sets the checksum of a message.

Description

This routine calculates the checksum of a 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 messages. It may be a checksum, a CRC, or some other algorithm.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a checksum field, then this routine will return `CFE_MSG_WRONG_MSG_TYPE`

Parameters

<code>in, out</code>	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
----------------------	---------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.
<code>CFE_MSG_WRONG_MSG_TYPE</code>	Error - wrong type.

```
10.31.2.2 CFE_MSG_GetFcnCode() CFE_Status_t CFE_MSG_GetFcnCode (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_FcnCode_t * FcnCode )
```

Gets the function code field from a message.

Description

This routine gets the function code from a message.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a function code field, then this routine will set FcnCode to zero and return [CFE_MSG_WRONG_MSG_TYPE](#)

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>FcnCode</i>	The function code from the message (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.
CFE_MSG_WRONG_MSG_TYPE	Error - wrong type.

Referenced by CF_AppPipe(), and CF_ProcessGroundCommand().

```
10.31.2.3 CFE_MSG_GetMsgTime() CFE_Status_t CFE_MSG_GetMsgTime (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_TIME_SysTime_t * Time )
```

Gets the time field from a message.

Description

This routine gets the time from a message.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a time field, then this routine will set Time to zero and return [CFE_MSG_WRONG_MSG_TYPE](#)
- Note default implementation of command messages do not have a time field.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Time</i>	Time from the message (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.
<i>CFE_MSG_WRONG_MSG_TYPE</i>	Error - wrong type.

10.31.2.4 CFE_MSG_SetFcnCode() [*CFE_Status_t*](#) CFE_MSG_SetFcnCode (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_FcnCode_t FcnCode )
```

Sets the function code field in a message.

Description

This routine sets the function code of a message.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a function code field, then this routine will do nothing to the message contents and will return [CFE_MSG_WRONG_MSG_TYPE](#).

Parameters

in,out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>FcnCode</i>	The function code to include in the message.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.
<i>CFE_MSG_WRONG_MSG_TYPE</i>	Error - wrong type.

10.31.2.5 CFE_MSG_SetMsgTime() [*CFE_Status_t*](#) CFE_MSG_SetMsgTime (

```
CFE_MSG_Message_t * MsgPtr,  
CFE_TIME_SysTime_t NewTime )
```

Sets the time field in a message.

Description

This routine sets the time of a message. Most applications will want to use [CFE_SB_TimeStampMsg](#) instead of this function. But, when needed, this API can be used to set multiple messages with identical time stamps.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a time field, then this routine will do nothing to the message contents and will return [CFE_MSG_WRONG_MSG_TYPE](#).
- Note default implementation of command messages do not have a time field.

Parameters

in, out	<i>MsgPtr</i>	A pointer to the message (must not be null).
in	<i>NewTime</i>	The time to include in the message. This will usually be a time from CFE_TIME_GetTime .

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.
CFE_MSG_WRONG_MSG_TYPE	Error - wrong type.

Referenced by CF_CFDP_Send().

10.31.2.6 CFE_MSG_ValidateChecksum() [CFE_Status_t](#) CFE_MSG_ValidateChecksum (

```
const CFE_MSG_Message_t * MsgPtr,  
bool * IsValid )
```

Validates the checksum of a message.

Description

This routine validates the checksum of a message according to an implementation-defined algorithm.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a checksum field, then this routine will return [CFE_MSG_WRONG_MSG_TYPE](#) and set the IsValid parameter false.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null). This must point to the first byte of the message header.
----	---------------	--

Parameters

<code>out</code>	<code>isValid</code>	Checksum validation result (must not be null) <ul style="list-style-type: none">• true - valid• false - invalid or not supported/implemented
------------------	----------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.
<code>CFE_MSG_WRONG_MSG_TYPE</code>	Error - wrong type.

10.32 cFE Message Id APIs

Functions

- `CFE_Status_t CFE_MSG_GetMsgId (const CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t *MsgId)`
Gets the message id from a message.
- `CFE_Status_t CFE_MSG_SetMsgId (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId)`
Sets the message id bits in a message.
- `CFE_Status_t CFE_MSG.GetTypeFromMsgId (CFE_SB_MsgId_t MsgId, CFE_MSG_Type_t *Type)`
Gets message type using message ID.

10.32.1 Detailed Description

10.32.2 Function Documentation

10.32.2.1 CFE_MSG_GetMsgId() `CFE_Status_t CFE_MSG_GetMsgId (`

```
    const CFE_MSG_Message_t * MsgPtr,  
    CFE_SB_MsgId_t * MsgId )
```

Gets the message id from a message.

Description

This routine gets the message id from a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>MsgId</i>	Message id (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.

Referenced by CF_AppPipe().

10.32.2.2 CFE_MSG.GetTypeFromMsgId() `CFE_Status_t CFE_MSG.GetTypeFromMsgId (`

```
    CFE_SB_MsgId_t MsgId,  
    CFE_MSG_Type_t * Type )
```

Gets message type using message ID.

Description

This routine gets the message type using the message ID

Parameters

in	<i>MsgId</i>	Message id
out	<i>Type</i>	Message type (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

```
10.32.2.3 CFE_MSG_SetMsgId() CFE_Status_t CFE_MSG_SetMsgId (
    CFE_MSG_Message_t * MsgPtr,
    CFE_SB_MsgId_t MsgId )
```

Sets the message id bits in a message.

Description

This routine sets the message id bits in a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

Note

This API only sets the bits in the header that make up the message ID. No other values in the header are modified.

The user should ensure that this function is only called with a valid MsgId parameter value. If called with an invalid value, the results are implementation-defined. The implementation may or may not return the error code [CFE_MSG_BAD_ARGUMENT](#) in this case.

Parameters

in,out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>MsgId</i>	Message id

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.33 cFE Message Integrity APIs

Functions

- `CFE_Status_t CFE_MSG_OriginationAction (CFE_MSG_Message_t *MsgPtr, size_t BufferSize, bool *IsAcceptable)`
Perform any necessary actions on a newly-created message, prior to sending.
- `CFE_Status_t CFE_MSG_VerificationAction (const CFE_MSG_Message_t *MsgPtr, size_t BufferSize, bool *IsAcceptable)`
Checks message integrity/acceptability.

10.33.1 Detailed Description

10.33.2 Function Documentation

10.33.2.1 `CFE_MSG_OriginationAction()`

```
CFE_Status_t CFE_MSG_OriginationAction (
    CFE_MSG_Message_t * MsgPtr,
    size_t BufferSize,
    bool * IsAcceptable )
```

Perform any necessary actions on a newly-created message, prior to sending.

Description

This routine updates and/or appends any necessary fields on a message, is invoked via SB just prior to broadcasting the message. The actions include updating any values that should be computed/updated per message, including:

- setting the sequence number
- updating the timestamp, if present
- computing any error control or checksum fields, if present

The MSG module implementation determines which header fields meet this criteria and how they should be computed. The BufferSize parameter indicates the allocation size message of the buffer that holds the message (i.e. the message envelope size). In some implementations, the allocated buffer may include extra space in order to append a CRC or digital signature.

See also

[CFE_MSG_VerificationAction](#)

Parameters

<code>in, out</code>	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
<code>in</code>	<code>BufferSize</code>	The size of the buffer encapsulating the message
<code>out</code>	<code>IsAcceptable</code>	Output variable to be set, indicates message acceptability (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Return values

CFE_MSG_BAD_ARGUMENT	Error - bad argument.
--------------------------------------	-----------------------

10.33.2.2 CFE_MSG_VerificationAction() [CFE_Status_t](#) CFE_MSG_VerificationAction (const [CFE_MSG_Message_t](#) * *MsgPtr*, size_t *BufferSize*, bool * *IsAcceptable*)

Checks message integrity/acceptability.

Description

This routine validates that any error-control field(s) in the message header matches the expected value.

The specific function of this API is entirely dependent on the header fields and may be a no-op if no error checking is implemented. In that case, it will always output "true".

Note

Due to the fact that software bus uses a multicast architecture, this function must not modify the message, as the buffer may be shared among multiple receivers. This should generally be the inverse of [CFE_MSG_OriginationAction\(\)](#), but on the origination side it may update header fields and/or modify the message, on the verification/receive side it must only check those fields, not modify them.

See also

[CFE_MSG_OriginationAction](#)

Parameters

in	<i>MsgPtr</i>	Message Pointer (must not be null)
in	<i>BufferSize</i>	The size of the buffer encapsulating the message
out	<i>IsAcceptable</i>	Output variable to be set, indicates message acceptability (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.34 cFE Pipe Management APIs

Functions

- `CFE_Status_t CFE_SB_CreatePipe (CFE_SB_Pipeld_t *PipeldPtr, uint16 Depth, const char *PipeName)`
Creates a new software bus pipe.
- `CFE_Status_t CFE_SB_DeletePipe (CFE_SB_Pipeld_t Pipeld)`
Delete a software bus pipe.
- `CFE_Status_t CFE_SB_Pipeld_ToIndex (CFE_SB_Pipeld_t Pipeld, uint32 *Idx)`
Obtain an index value correlating to an SB Pipe ID.
- `CFE_Status_t CFE_SB_SetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 Opts)`
Set options on a pipe.
- `CFE_Status_t CFE_SB_GetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 *OptsPtr)`
Get options on a pipe.
- `CFE_Status_t CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_Pipeld_t Pipeld)`
Get the pipe name for a given id.
- `CFE_Status_t CFE_SB_GetPipeldByName (CFE_SB_Pipeld_t *PipeldPtr, const char *PipeName)`
Get pipe id by pipe name.

10.34.1 Detailed Description

10.34.2 Function Documentation

10.34.2.1 CFE_SB_CreatePipe() `CFE_Status_t CFE_SB_CreatePipe (`
`CFE_SB_Pipeld_t * PipeIdPtr,`
`uint16 Depth,`
`const char * PipeName)`

Creates a new software bus pipe.

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

out	<code>PipeldPtr</code>	A pointer to a variable of type <code>CFE_SB_Pipeld_t</code> (must not be null), which will be filled in with the pipe ID information by the <code>CFE_SB_CreatePipe</code> routine. <code>*PipeldPtr</code> is the identifier for the created pipe.
in	<code>Depth</code>	The maximum number of messages that will be allowed on this pipe at one time.
in	<code>PipeName</code>	A string (must not be null) to be used to identify this pipe in error messages and routing information telemetry. The string must be no longer than <code>OS_MAX_API_NAME</code> (including terminator). Longer strings will be truncated.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_MAX_PIPES_MET	Max Pipes Met.
CFE_SB_PIPE_CR_ERR	Pipe Create Error.

See also

[CFE_SB_DeletePipe](#) [CFE_SB_GetPipeOpts](#) [CFE_SB_SetPipeOpts](#) [CFE_SB_GetPipeIdByName](#)

Referenced by [CF_AppInit\(\)](#), and [CF_CFDP_InitEngine\(\)](#).

10.34.2.2 CFE_SB_DeletePipe() [CFE_Status_t](#) CFE_SB_DeletePipe ([CFE_SB_PipeId_t](#) PipeId)

Delete a software bus pipe.

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	<i>PipeId</i>	The pipe ID (obtained previously from CFE_SB_CreatePipe) of the pipe to be deleted.
----	---------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_CreatePipe](#) [CFE_SB_GetPipeOpts](#) [CFE_SB_SetPipeOpts](#) [CFE_SB_GetPipeIdByName](#)

Referenced by [CF_CFDP_DisableEngine\(\)](#).

10.34.2.3 CFE_SB_GetPipeIdByName() [`CFE_Status_t CFE_SB_GetPipeIdByName \(`](#)

```
CFE\_SB\_PipeId\_t \* PipeIdPtr,
const char \* PipeName \)
```

Get pipe id by pipe name.

Description

This routine finds the pipe id for a pipe name.

Parameters

in	<i>PipeName</i>	The name of the pipe (must not be null).
out	<i>PipeIdPtr</i>	The PipeId for that name (must not be null).

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_CreatePipe](#) [CFE_SB_DeletePipe](#) [CFE_SB_SetPipeOpts](#) [CFE_SB_PIPEOPTS_IGNOREMINE](#)

10.34.2.4 CFE_SB_GetPipeName() [`CFE_Status_t CFE_SB_GetPipeName \(`](#)

```
char \* PipeNameBuf,
size\_t PipeNameSize,
CFE\_SB\_PipeId\_t PipeId \)
```

Get the pipe name for a given id.

Description

This routine finds the pipe name for a pipe id.

Parameters

out	<i>PipeNameBuf</i>	The buffer to receive the pipe name (must not be null).
in	<i>PipeNameSize</i>	The size (in chars) of the PipeName buffer (must not be zero).
in	<i>PipeId</i>	The PipeId for that name.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_CreatePipe](#) [CFE_SB_DeletePipe](#) [CFE_SB_SetPipeOpts](#) [CFE_SB_GetPipeIdByName](#)

10.34.2.5 CFE_SB_GetPipeOpts() [CFE_Status_t](#) CFE_SB_GetPipeOpts (
 [CFE_SB_PipeId_t](#) *PipeId*,
 [uint8](#) * *OptsPtr*)

Get options on a pipe.

Description

This routine gets the current options on a pipe.

Parameters

in	<i>PipeId</i>	The pipe ID of the pipe to get options from.
out	<i>OptsPtr</i>	A bit field of options: cFE SB Pipe options (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_CreatePipe](#) [CFE_SB_DeletePipe](#) [CFE_SB_SetPipeOpts](#) [CFE_SB_GetPipeIdByName](#) [CFE_SB_PIPEOPTS_IGNOREMIN](#)

10.34.2.6 CFE_SB_PipeId_ToIndex() [CFE_Status_t](#) CFE_SB_PipeId_ToIndex (
 [CFE_SB_PipeId_t](#) *PipeID*,
 [uint32](#) * *Idx*)

Obtain an index value correlating to an SB Pipe ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of a pipe ID and an app ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

Note

There is no inverse of this function - indices cannot be converted back to the original PipeID value. The caller should retain the original ID for future use.

Parameters

in	<i>PipeID</i>	Pipe ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

10.34.2.7 CFE_SB_SetPipeOpts() [CFE_Status_t](#) CFE_SB_SetPipeOpts (
 [CFE_SB_PipeId_t](#) *PipeId*,
 [uint8](#) *Opts*)

Set options on a pipe.

Description

This routine sets (or clears) options to alter the pipe's behavior. Options are (re)set every call to this routine.

Parameters

in	<i>PipeId</i>	The pipe ID of the pipe to set options on.
in	<i>Opts</i>	A bit field of options: cFE SB Pipe options

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_CreatePipe](#) [CFE_SB_DeletePipe](#) [CFE_SB_GetPipeOpts](#) [CFE_SB_GetPipeIdByName](#) [CFE_SB_PIPEOPTS_IGNOREMIN](#)

10.35 cFE Message Subscription Control APIs

Functions

- [CFE_Status_t 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.
- [CFE_Status_t CFE_SB_Subscribe \(CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId\)](#)
Subscribe to a message on the software bus with default parameters.
- [CFE_Status_t 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.
- [CFE_Status_t CFE_SB_Unsubscribe \(CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId\)](#)
Remove a subscription to a message on the software bus.
- [CFE_Status_t 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.

10.35.1 Detailed Description

10.35.2 Function Documentation

10.35.2.1 CFE_SB_Subscribe() [CFE_Status_t CFE_SB_Subscribe \(](#)
`CFE_SB_MsgId_t MsgId,`
`CFE_SB_PipeId_t PipeId)`

Subscribe to a message on the software bus with default parameters.

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	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_MAX_MSGS_MET</i>	(return value only verified in coverage test) Max Messages Met.
<i>CFE_SB_MAX_DESTS_MET</i>	Max Destinations Met.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

See also

[CFE_SB_SubscribeEx](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_Unsubscribe](#), [CFE_SB_UnsubscribeLocal](#)

Referenced by CF_AppInit().

10.35.2.2 CFE_SB_SubscribeEx() *CFE_Status_t* 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.

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	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
in	<i>Quality</i>	The requested Quality of Service (QoS) required of the messages. Most callers will use CFE_SB_DEFAULT_QOS for this parameter.
in	<i>MsgLim</i>	The maximum number of messages with this Message ID to allow in this pipe at the same time.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_MAX_MSGS_MET</i>	(return value only verified in coverage test) Max Messages Met.
<i>CFE_SB_MAX_DESTS_MET</i>	Max Destinations Met.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

See also

[CFE_SB_Subscribe](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_Unsubscribe](#), [CFE_SB_UnsubscribeLocal](#)

10.35.2.3 CFE_SB_SubscribeLocal() `CFE_Status_t 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.

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.

Software Bus Network (SBN) application is an example use case, where local subscriptions should not be reported to peers.

Assumptions, External Events, and Notes:

- This API is typically only used by Software Bus Network (SBN) Application

Parameters

in	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
in	<i>MsgLim</i>	The maximum number of messages with this Message ID to allow in this pipe at the same time.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_MAX_MSGS_MET	(return value only verified in coverage test) Max Messages Met.
CFE_SB_MAX_DESTS_MET	Max Destinations Met.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_BUF_ALOC_ERR	(return value only verified in coverage test) Buffer Allocation Error.

See also

[CFE_SB_Subscribe](#), [CFE_SB_SubscribeEx](#), [CFE_SB_Unsubscribe](#), [CFE_SB_UnsubscribeLocal](#)

Referenced by CF_CFDP_InitEngine().

10.35.2.4 CFE_SB_Unsubscribe() `CFE_Status_t CFE_SB_Unsubscribe (`

```
    CFE_SB_MsgId_t MsgId,
    CFE_SB_PipeId_t PipeId )
```

Remove a subscription to a message on the software bus.

Description

This routine removes the specified pipe from the destination list for the specified message ID.

Assumptions, External Events, and Notes:

If the Pipe is not subscribed to MsgId, the CFE_SB_UNSUB_NO_SUBS_EID event will be generated and [CFE_SUCCESS](#) will be returned

Parameters

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_Subscribe](#), [CFE_SB_SubscribeEx](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_UnsubscribeLocal](#)

10.35.2.5 CFE_SB_UnsubscribeLocal() [CFE_Status_t](#) 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.

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. If the Pipe is not subscribed to MsgId, the CFE_SB_UNSUB_NO_SUBS_EID event will be generated and [CFE_SUCCESS](#) will be returned

Parameters

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_SB_Subscribe](#), [CFE_SB_SubscribeEx](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_Unsubscribe](#)

10.36 cFE Send/Receive Message APIs

Functions

- `CFE_Status_t CFE_SB_TransmitMsg (const CFE_MSG_Message_t *MsgPtr, bool IsOrigination)`
Transmit a message.
- `CFE_Status_t CFE_SB_ReceiveBuffer (CFE_SB_Buffer_t **BufPtr, CFE_SB_PipeId_t PipeId, int32 TimeOut)`
Receive a message from a software bus pipe.

10.36.1 Detailed Description

10.36.2 Function Documentation

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

Receive a message from a software bus pipe.

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, out	<code>BufPtr</code>	A pointer to the software bus buffer to receive to (must not be null). Typically a caller declares a ptr of type <code>CFE_SB_Buffer_t</code> (i.e. <code>CFE_SB_Buffer_t *Ptr</code>) then gives the address of that pointer (<code>&Ptr</code>) as this parameter. After a successful receipt of a message, <code>*BufPtr</code> will point to the first byte of the software bus buffer. 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 <code>*BufPtr</code> is valid only until the next call to <code>CFE_SB_ReceiveBuffer</code> for the same pipe.
in	<code>PipeId</code>	The pipe ID of the pipe containing the message to be obtained.
in	<code>TimeOut</code>	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 <code>CFE_SB_POLL</code> for a non-blocking receive or <code>CFE_SB_PEND_FOREVER</code> to wait forever for a message to arrive.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_SB_BAD_ARGUMENT</code>	Bad Argument.

Return values

CFE_SB_TIME_OUT	Time Out.
CFE_SB_PIPE_RD_ERR	(return value only verified in coverage test) Pipe Read Error.
CFE_SB_NO_MESSAGE	No Message.

Referenced by CF_AppMain(), and CF_CFDP_ReceiveMessage().

```
10.36.2.2 CFE_SB_TransmitMsg() CFE\_Status\_t CFE_SB_TransmitMsg (
    const CFE\_MSG\_Message\_t * MsgPtr,
    bool IsOrigination )
```

Transmit a message.

Description

This routine copies the specified message into a software bus buffer which is then transmitted to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message.

The *IsOrigination* parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. This enables the message origination actions as determined by the CFE MSG module, which may include (but not limited to):

- Updating sequence number
- Updating timestamp
- Calculating a CRC, checksum, or other message error control field

Conversely, when forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

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 returning control to the caller.
- In previous versions of CFE, the boolean parameter referred to the sequence number header of telemetry messages only. This has been extended to apply more generically to any headers, as determined by the CFE MSG implementation.

Parameters

in	<i>MsgPtr</i>	A pointer to the message to be sent (must not be null). This must point to the first byte of the message header.
in	<i>IsOrigination</i>	Update the headers of the message

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_MSG_TOO_BIG</i>	Message Too Big.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

Referenced by CF_SendHkCmd().

10.37 cFE Zero Copy APIs

Functions

- [CFE_SB_Buffer_t * CFE_SB_AllocateMessageBuffer \(size_t MsgSize\)](#)
Get a buffer pointer to use for "zero copy" SB sends.
- [CFE_Status_t CFE_SB_ReleaseMessageBuffer \(CFE_SB_Buffer_t *BufPtr\)](#)
Release an unused "zero copy" buffer pointer.
- [CFE_Status_t CFE_SB_TransmitBuffer \(CFE_SB_Buffer_t *BufPtr, bool IsOrigination\)](#)
Transmit a buffer.

10.37.1 Detailed Description

10.37.2 Function Documentation

10.37.2.1 CFE_SB_AllocateMessageBuffer() [CFE_SB_Buffer_t*](#) CFE_SB_AllocateMessageBuffer ([size_t](#) *MsgSize*)

Get a buffer pointer to use for "zero copy" SB sends.

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_TransmitBuffer\(\)](#) function. This interface avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer.

Assumptions, External Events, and Notes:

1. The pointer returned by [CFE_SB_AllocateMessageBuffer\(\)](#) is only good for one call to [CFE_SB_TransmitBuffer\(\)](#).
2. Once a buffer has been successfully transmitted (as indicated by a successful return from [CFE_SB_TransmitBuffer\(\)](#)) the buffer becomes owned by the SB application. It will automatically be freed by SB once all recipients have finished reading it.
3. Applications must not de-reference the message pointer (for reading or writing) after the call to [CFE_SB_TransmitBuffer\(\)](#).
4. If [CFE_SB_ReleaseMessageBuffer](#) should be used only if a message is not transmitted

Parameters

in	<i>MsgSize</i>	The size of the SB message buffer the caller wants (including the SB message header).
--------------------	----------------	---

Returns

A pointer to a memory buffer that message data can be written to for use with [CFE_SB_TransmitBuffer\(\)](#).

Referenced by CF_CFDP_MsgOutGet(), and CF_CFDP_SendEotPkt().

10.37.2.2 CFE_SB_ReleaseMessageBuffer() [CFE_Status_t](#) CFE_SB_ReleaseMessageBuffer ([CFE_SB_Buffer_t](#) * *BufPtr*)

Release an unused "zero copy" buffer pointer.

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_AllocateMessageBuffer\(\)](#), but (due to some error condition) never uses that pointer in a call to [CFE_SB_TransmitBuffer\(\)](#).

Parameters

in	<i>BufPtr</i>	A pointer to the SB internal buffer (must not be null). This must be a pointer returned by a call to CFE_SB_AllocateMessageBuffer() , but never used in a call to CFE_SB_TransmitBuffer() .
----	---------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BUFFER_INVALID	Buffer Invalid.

Referenced by CF_CFDP_MsgOutGet().

10.37.2.3 CFE_SB_TransmitBuffer() [CFE_Status_t](#) CFE_SB_TransmitBuffer (
 [CFE_SB_Buffer_t](#) * *BufPtr*,
 bool *IsOrigination*)

Transmit a buffer.

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_AllocateMessageBuffer\(\)](#)). This interface is more complicated than the normal [CFE_SB_TransmitMsg](#) 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.

The *IsOrigination* parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. This enables the message origination actions as determined by the CFE MSG module, which may include (but not limited to):

- Updating sequence number
- Updating timestamp
- Calculating a CRC, checksum, or other message error control field

Conversely, when forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

Assumptions, External Events, and Notes:

1. A handle returned by [CFE_SB_AllocateMessageBuffer](#) is "consumed" by a *successful* call to [CFE_SB_TransmitBuffer](#).
2. If this function returns CFE_SUCCESS, this indicates the zero copy handle is now owned by software bus, and is no longer owned by the calling application, and should not be re-used.
3. However if this function fails (returns any error status) it does not change the state of the buffer at all, meaning the calling application still owns it. (a failure means the buffer is left in the same state it was before the call).
4. Applications should be written as if [CFE_SB_AllocateMessageBuffer](#) is equivalent to a `malloc()` and a successful call to [CFE_SB_TransmitBuffer](#) is equivalent to a `free()`.
5. Applications must not de-reference the message pointer (for reading or writing) after a successful call to [CFE_SB_TransmitBuffer](#).
6. This function will increment and apply the internally tracked sequence counter if set to do so.

Parameters

in	<i>BufPtr</i>	A pointer to the buffer to be sent (must not be null).
in	<i>IsOrigination</i>	Update applicable header field(s) of a newly constructed message

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_MSG_TOO_BIG	Message Too Big.

Referenced by CF_CFDP_Send(), and CF_CFDP_SendEotPkt().

10.38 cFE Message Characteristics APIs

Functions

- void `CFE_SB_SetUserDataLength (CFE_MSG_Message_t *MsgPtr, size_t DataLength)`
Sets the length of user data in a software bus message.
- void `CFE_SB_TimeStampMsg (CFE_MSG_Message_t *MsgPtr)`
Sets the time field in a software bus message with the current spacecraft time.
- int32 `CFE_SB_MessageStringSet (char *DestStringPtr, const char *SourceStringPtr, size_t DestMaxSize, size_t SourceMaxSize)`
Copies a string into a software bus message.
- void * `CFE_SB_GetUserData (CFE_MSG_Message_t *MsgPtr)`
Get a pointer to the user data portion of a software bus message.
- size_t `CFE_SB_GetUserDataLength (const CFE_MSG_Message_t *MsgPtr)`
Gets the length of user data in a software bus message.
- int32 `CFE_SB_MessageStringGet (char *DestStringPtr, const char *SourceStringPtr, const char *DefaultString, size_t DestMaxSize, size_t SourceMaxSize)`
Copies a string out of a software bus message.

10.38.1 Detailed Description

10.38.2 Function Documentation

10.38.2.1 CFE_SB_GetUserData() `void* CFE_SB_GetUserData (` `CFE_MSG_Message_t * MsgPtr)`

Get a pointer to the user data portion of a software bus message.

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	<code>MsgPtr</code>	A pointer to the buffer that contains the software bus message (must not be null).
----	---------------------	--

Returns

A pointer to the first byte of user data within the software bus message.

10.38.2.2 CFE_SB_GetUserDataLength() `size_t CFE_SB_GetUserDataLength (` `const CFE_MSG_Message_t * MsgPtr)`

Gets the length of user data in a software bus message.

Description

This routine returns the size of the user data in a software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

Returns

The size (in bytes) of the user data in the software bus message.

Return values

0	if an error occurs, such as if the <i>MsgPtr</i> argument is not valid.
---	---

```
10.38.2.3 CFE_SB_MessageStringGet() int32 CFE_SB_MessageStringGet (
    char * DestStringPtr,
    const char * SourceStringPtr,
    const char * DefaultString,
    size_t DestMaxSize,
    size_t SourceMaxSize )
```

Copies a string out of a software bus message.

Description

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This function should replace use of C library functions such as strcpy/strncpy when copying strings out of software bus messages to local storage buffers.

Up to [SourceMaxSize] or [DestMaxSize-1] (whichever is smaller) characters will be copied from the source buffer to the destination buffer, and a NUL termination character will be written to the destination buffer as the last character.

If the *DefaultString* pointer is non-NULL, it will be used in place of the source string if the source is an empty string. This is typically a string constant that comes from the platform configuration, allowing default values to be assumed for fields that are unspecified.

IMPORTANT - the default string, if specified, must be null terminated. This will be the case if a string literal is passed in (the typical/expected use case).

If the default is NULL, then only the source string will be copied, and the result will be an empty string if the source was empty.

If the destination buffer is too small to store the entire string, it will be truncated, but it will still be null terminated.

Parameters

out	<i>DestStringPtr</i>	Pointer to destination buffer (must not be null)
-----	----------------------	--

Parameters

in	<i>SourceStringPtr</i>	Pointer to source buffer (component of SB message definition)
in	<i>DefaultString</i>	Default string to use if source is empty
in	<i>DestMaxSize</i>	Size of destination storage buffer (must not be zero)
in	<i>SourceMaxSize</i>	Size of source buffer as defined by the message definition

Returns

Number of characters copied or error code, see [cFE Return Code Defines](#)

Return values

CFE_SB_BAD_ARGUMENT	Bad Argument.
-------------------------------------	---------------

10.38.2.4 CFE_SB_MessageStringSet() `int32 CFE_SB_MessageStringSet (`
 `char * DestStringPtr,`
 `const char * SourceStringPtr,`
 `size_t DestMaxSize,`
 `size_t SourceMaxSize)`

Copies a string into a software bus message.

Description

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This performs a very similar function to "strncpy()" except that the sizes of *both* buffers are passed in. Neither buffer is required to be null-terminated, but copying will stop after the first termination character is encountered. If the destination buffer is not completely filled by the source data (such as if the supplied string was shorter than the allotted length) the destination buffer will be padded with NUL characters up to the size of the buffer, similar to what strncpy() does. This ensures that the entire destination buffer is set.

Note

If the source string buffer is already guaranteed to be null terminated, then there is no difference between the C library "strncpy()" function and this implementation. It is only necessary to use this when termination of the source buffer is not guaranteed.

Parameters

out	<i>DestStringPtr</i>	Pointer to destination buffer (component of SB message definition) (must not be null)
in	<i>SourceStringPtr</i>	Pointer to source buffer (must not be null)
in	<i>DestMaxSize</i>	Size of destination buffer as defined by the message definition
in	<i>SourceMaxSize</i>	Size of source buffer

Returns

Number of characters copied or error code, see [cFE Return Code Defines](#)

Return values

CFE_SB_BAD_ARGUMENT	Bad Argument.
-------------------------------------	---------------

10.38.2.5 CFE_SB_SetUserDataLength() `void CFE_SB_SetUserDataLength (`

```
    CFE_MSG_Message_t * MsgPtr,
    size_t DataLength )
```

Sets the length of user data in a software bus message.

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	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
in	<i>DataLength</i>	The length to set (size of the user data, in bytes).

10.38.2.6 CFE_SB_TimeStampMsg() `void CFE_SB_TimeStampMsg (`

```
    CFE_MSG_Message_t * MsgPtr )
```

Sets the time field in a software bus message with the current spacecraft time.

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	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

Referenced by CF_CFDP_SendEotPkt(), and CF_SendHkCmd().

10.39 cFE Message ID APIs

Functions

- bool [CFE_SB_IsValidMsgId](#) ([CFE_SB_MsgId_t](#) MsgId)
Identifies whether a given [CFE_SB_MsgId_t](#) is valid.
- static bool [CFE_SB_MsgId_Equal](#) ([CFE_SB_MsgId_t](#) MsgId1, [CFE_SB_MsgId_t](#) MsgId2)
Identifies whether 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_Atom_t](#) [CFE_SB_ValueToMsgId](#) ([CFE_SB_MsgId_Atom_t](#) MsgIdValue)
Converts a normal integer into a [CFE_SB_MsgId_t](#).
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_CmdTopicIdToMsgId](#) ([uint16](#) TopicId, [uint16](#) InstanceNum)
Converts a topic ID and instance number combination into a MsgID value integer.
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_TlmTopicIdToMsgId](#) ([uint16](#) TopicId, [uint16](#) InstanceNum)
Converts a topic ID and instance number combination into a MsgID value integer.
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_GlobalCmdTopicIdToMsgId](#) ([uint16](#) TopicId)
Converts a topic ID to a MsgID value integer for Global commands.
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_GlobalTlmTopicIdToMsgId](#) ([uint16](#) TopicId)
Converts a topic ID to a MsgID value integer for Global telemetry.
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_LocalCmdTopicIdToMsgId](#) ([uint16](#) TopicId)
Converts a topic ID to a MsgID value integer for local commands.
- [CFE_SB_MsgId_Atom_t](#) [CFE_SB_LocalTlmTopicIdToMsgId](#) ([uint16](#) TopicId)
Converts a topic ID to a MsgID value integer for local telemetry.

10.39.1 Detailed Description

10.39.2 Function Documentation

10.39.2.1 [CFE_SB_CmdTopicIdToMsgId\(\)](#) [CFE_SB_MsgId_Atom_t](#) [CFE_SB_CmdTopicIdToMsgId](#) ([uint16](#) TopicId, [uint16](#) InstanceNum)

Converts a topic ID and instance number combination into a MsgID value integer.

Description

This function accepts a data pair of topic ID + instance number and returns the corresponding MsgID Value (integer) for commands.

Assumptions and Notes:

A topic ID identifies a certain data stream from an application, for example the CFE Software bus ground commands ([CFE_MISSION_SB_CMD_TOPICID](#)). In contrast to MsgID, the topic ID is consistent across all CPUs in a system, whereas each CPU instance will have a unique MsgID.

CPU instance numbers are 1-based. The instance number of 0 is reserved for "global" MsgID values that are the same on all CPUs. The PSP function may be used to obtain the current CPU number for the host processor.

See also

[CFE_SB_TlmTopicIdToMsgId\(\)](#), [CFE_PSP_GetProcessorId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.2 CFE_SB_GlobalCmdTopicIdToMsgId() `CFE_SB_MsgId_Atom_t` `CFE_SB_GlobalCmdTopicIdToMsgId (`
 `uint16 TopicId)`

Converts a topic ID to a MsgID value integer for Global commands.

Description

This is a wrapper around [CFE_SB_CmdTopicIdToMsgId\(\)](#) for topic IDs which are the same on all CPUs within a system (i.e. not specific to a certain processor)

Assumptions and Notes:

Global MsgIDs may be used when only a single instance of a service exists within the system. The CFE framework does not use this feature for commands, but is defined for future use.

See also

[CFE_SB_CmdTopicIdToMsgId\(\)](#), [CFE_SB_LocalCmdTopicIdToMsgId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.3 CFE_SB_GlobalTlmTopicIdToMsgId() `CFE_SB_MsgId_Atom_t` `CFE_SB_GlobalTlmTopicIdToMsgId (`
 `uint16 TopicId)`

Converts a topic ID to a MsgID value integer for Global telemetry.

Description

This is a wrapper around [CFE_SB_TlmTopicIdToMsgId\(\)](#) for topic IDs which are the same on all CPUs within a system (i.e. not specific to a certain processor)

Assumptions and Notes:

Global MsgIDs may be used when only a single instance of a service exists within the system. An example for such telemetry is the time synchronization service published by CFE_TIME.

See also

[CFE_SB_TlmTopicIdToMsgId\(\)](#), [CFE_SB_LocalTlmTopicIdToMsgId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.4 CFE_SB_IsValidMsgId() `bool` `CFE_SB_IsValidMsgId (`
 `CFE_SB_MsgId_t MsgId)`

Identifies whether a given [CFE_SB_MsgId_t](#) is valid.

Description

Implements a basic sanity check on the value provided

Returns

Boolean message ID validity indicator

Return values

<i>true</i>	Message ID is within the valid range
<i>false</i>	Message ID is not within the valid range

10.39.2.5 CFE_SB_LocalCmdTopicIdToMsgId() *CFE_SB_MsgId_Atom_t* CFE_SB_LocalCmdTopicIdToMsgId (*uint16 TopicId*)

Converts a topic ID to a MsgID value integer for local commands.

Description

This is a wrapper around [CFE_SB_CmdTopicIdToMsgId\(\)](#) for topic IDs which are unique on all CPUs within a system (i.e. specific to a certain processor)

Assumptions and Notes:

This assumes the caller is referring to a service running on the same processor instance as itself.

See also

[CFE_SB_CmdTopicIdToMsgId\(\)](#), [CFE_SB_LocalTlmTopicIdToMsgId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.6 CFE_SB_LocalTlmTopicIdToMsgId() *CFE_SB_MsgId_Atom_t* CFE_SB_LocalTlmTopicIdToMsgId (*uint16 TopicId*)

Converts a topic ID to a MsgID value integer for local telemetry.

Description

This is a wrapper around [CFE_SB_TlmTopicIdToMsgId\(\)](#) for topic IDs which are unique on all CPUs within a system (i.e. specific to a certain processor)

Assumptions and Notes:

This assumes the caller is referring to a service running on the same processor instance as itself.

See also

[CFE_SB_TlmTopicIdToMsgId\(\)](#), [CFE_SB_LocalCmdTopicIdToMsgId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.7 CFE_SB_MsgId_Equal() static bool CFE_SB_MsgId_Equal (
 CFE_SB_MsgId_t MsgId1,
 CFE_SB_MsgId_t MsgId2) [inline], [static]

Identifies whether two [CFE_SB_MsgId_t](#) values are equal.

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

Boolean message ID equality indicator

Return values

<i>true</i>	Message IDs are Equal
<i>false</i>	Message IDs are not Equal

Definition at line 791 of file cfe_sb.h.

References CFE_SB_MSGID_UNWRAP_VALUE.

10.39.2.8 CFE_SB_MsgIdToValue() static [CFE_SB_MsgId_Atom_t](#) CFE_SB_MsgIdToValue (
 [CFE_SB_MsgId_t](#) MsgId) [inline], [static]

Converts a [CFE_SB_MsgId_t](#) to a normal integer.

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 822 of file cfe_sb.h.

References CFE_SB_MSGID_UNWRAP_VALUE.

Referenced by CF_AppPipe().

10.39.2.9 CFE_SB_TlmTopicIdToMsgId() [CFE_SB_MsgId_Atom_t](#) CFE_SB_TlmTopicIdToMsgId (
 uint16 TopicId,
 uint16 InstanceNum)

Converts a topic ID and instance number combination into a MsgID value integer.

Description

This function accepts a data pair of topic ID + instance number and returns the corresponding MsgID Value (integer) for telemetry.

Assumptions and Notes:

A topic ID identifies a certain data stream from an application, for example the CFE Software bus housekeeping telemetry (CFE_MISSION_SB_HK_TLM_TOPICID). In contrast to MsgID, the topic ID is consistent across all CPUs in a system, whereas each CPU instance will have a unique MsgID.

CPU instance numbers are 1-based. The instance number of 0 is reserved for "global" MsgID values that are the same on all CPUs. The PSP function may be used to obtain the current CPU number for the host processor.

See also

[CFE_SB_CmdTopicIdToMsgId\(\)](#), [CFE_PSP_GetProcessorId\(\)](#)

Returns

Integer representation of the [CFE_SB_MsgId_t](#)

10.39.2.10 CFE_SB_ValueToMsgId() static [CFE_SB_MsgId_t](#) CFE_SB_ValueToMsgId ([CFE_SB_MsgId_Atom_t](#) MsgIdValue) [inline], [static]

Converts a normal integer into a [CFE_SB_MsgId_t](#).

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 851 of file cfe_sb.h.

References [CFE_SB_MSGID_C](#).

Referenced by [CF_AppInit\(\)](#), [CF_CFDP_InitEngine\(\)](#), [CF_CFDP_MsgOutGet\(\)](#), and [CF_CFDP_SendEotPkt\(\)](#).

10.40 cFE SB Pipe options

Macros

- #define CFE_SB_PIPEOPTS_IGNOREMINE 0x00000001

Messages sent by the app that owns this pipe will not be sent to this pipe.

10.40.1 Detailed Description

10.40.2 Macro Definition Documentation

10.40.2.1 CFE_SB_PIPEOPTS_IGNOREMINE #define CFE_SB_PIPEOPTS_IGNOREMINE 0x00000001

Messages sent by the app that owns this pipe will not be sent to this pipe.

Definition at line 131 of file cfe_sb_api_typedefs.h.

10.41 cFE Registration APIs

Functions

- `CFE_Status_t CFE_TBL_Register (CFE_TBL_Handle_t *TblHandlePtr, const char *Name, size_t Size, uint16 TblOptionFlags, CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr)`
Register a table with cFE to obtain Table Management Services.
- `CFE_Status_t CFE_TBL_Share (CFE_TBL_Handle_t *TblHandlePtr, const char *TblName)`
Obtain handle of table registered by another application.
- `CFE_Status_t CFE_TBL_Unregister (CFE_TBL_Handle_t TblHandle)`
Unregister a table.

10.41.1 Detailed Description

10.41.2 Function Documentation

10.41.2.1 CFE_TBL_Register() `CFE_Status_t CFE_TBL_Register (`
`CFE_TBL_Handle_t * TblHandlePtr,`
`const char * Name,`
`size_t Size,`
`uint16 TblOptionFlags,`
`CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr)`

Register a table with cFE to obtain Table Management Services.

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

out	<code>TblHandlePtr</code>	a pointer to a <code>CFE_TBL_Handle_t</code> type variable (must not be null) 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. <code>*TblHandlePtr</code> is the handle used to identify table to cFE when performing Table operations. This value is returned at address specified by <code>TblHandlePtr</code> .
in	<code>Name</code>	The raw table name. This name will be combined with the name of the application to produce a name of the form "AppName.RawTableName". This application specific name will be used in commands for modifying or viewing the contents of the table.
in	<code>Size</code>	The size, in bytes, of the table to be created (must not be zero). This is the size that will be allocated as a shared memory resource between the Table Management Service and the calling application.

Parameters

in	<i>TblOptionFlags</i>	<p>Flag bits indicating selected options for table. A bitwise OR of the following option flags:</p> <ul style="list-style-type: none"> • 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 ensure 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 quick and it could be blocked. Therefore, critical tables should not be

Parameters

in	<i>TblValidationFuncPtr</i>	<p>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:</p> <pre>int32 CallbackFunc(void *TblPtr);</pre> <p>where</p> <p>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.</p>
----	-----------------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_RECOVERED_TBL	Recovered Table.
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE	Duplicate Table With Different Size.
CFE_TBL_ERR_DUPLICATE_NOT OWNED	Duplicate Table And Not Owned.
CFE_TBL_ERR_REGISTRY_FULL	Registry Full.
CFE_TBL_ERR_HANDLES_FULL	Handles Full.
CFE_TBL_ERR_INVALID_SIZE	Invalid Size.
CFE_TBL_ERR_INVALID_NAME	Invalid Name.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_BAD_ARGUMENT	Bad Argument.
CFE_TBL_ERR_INVALID_OPTIONS	Invalid Options.
CFE_TBL_WARN_DUPLICATE	Duplicate Warning.
CFE_TBL_WARN_NOT_CRITICAL	Not Critical Warning.

See also

[CFE_TBL_Unregister](#), [CFE_TBL_Share](#)

Referenced by [CF_TableInit\(\)](#).

```
10.41.2.2 CFE_TBL_Share() CFE\_Status\_t CFE_TBL_Share (
    CFE\_TBL\_Handle\_t * TblHandlePtr,
    const char * TblName )
```

Obtain handle of table registered by another application.

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

<i>out</i>	<i>TblHandlePtr</i>	A pointer to a CFE_TBL_Handle_t type variable (must not be null) 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. *TblHandlePtr is the handle used to identify table to cFE when performing Table operations. This value is returned at the address specified by TblHandlePtr.
<i>in</i>	<i>TblName</i>	The application specific name of the table of the form "AppName.RawTableName", where RawTableName is the name specified in the CFE_TBL_Register API call. Example: "ACS.TamParams" for a table called "TamParams" that was registered by the application called "ACS".

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_ERR_HANDLES_FULL	Handles Full.
CFE_TBL_ERR_INVALID_NAME	Invalid Name.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

[CFE_TBL_Unregister](#), [CFE_TBL_Register](#)

10.41.2.3 CFE_TBL_Unregister()

```
CFE_Status_t CFE_TBL_Unregister (
    CFE_TBL_Handle_t TblHandle )
```

Unregister a table.

Description

When an application is being removed from the system, ES will clean up/free all the application related resources including tables so apps are not required to call this function.

A valid use-case for this API is to unregister a shared table if access is no longer needed or the owning application was removed from the system (CS app is an example).

Typically apps should only register tables during initialization and registration/unregistration by the owning application during operation should be avoided. If unavoidable, special care needs to be taken (especially for shared tables) to avoid race conditions due to competing requests from multiple tasks.

Note the table will not be removed from memory until all table access links have been removed (registration and all shared access).

Assumptions, External Events, and Notes:

None

Parameters

in	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be unregistered.
----	------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

[CFE_TBL_Share](#), [CFE_TBL_Register](#)

10.42 cFE Manage Table Content APIs

Functions

- `CFE_Status_t 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.
- `CFE_Status_t CFE_TBL_Update (CFE_TBL_Handle_t TblHandle)`
Update contents of a specified table, if an update is pending.
- `CFE_Status_t CFE_TBL_Validate (CFE_TBL_Handle_t TblHandle)`
Perform steps to validate the contents of a table image.
- `CFE_Status_t CFE_TBL_Manage (CFE_TBL_Handle_t TblHandle)`
Perform standard operations to maintain a table.
- `CFE_Status_t CFE_TBL_DumpToBuffer (CFE_TBL_Handle_t TblHandle)`
Copies the contents of a Dump Only Table to a shared buffer.
- `CFE_Status_t CFE_TBL_Modified (CFE_TBL_Handle_t TblHandle)`
Notify cFE Table Services that table contents have been modified by the Application.

10.42.1 Detailed Description

10.42.2 Function Documentation

10.42.2.1 `CFE_TBL_DumpToBuffer()` `CFE_Status_t CFE_TBL_DumpToBuffer (CFE_TBL_Handle_t TblHandle)`

Copies the contents of a Dump Only Table to a shared buffer.

Description

Typically, apps should just call `CFE_TBL_Manage` as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a dump should be performed.

Assumptions, External Events, and Notes:

If the table does not have a dump pending status, nothing will occur (no error, no dump)

Parameters

in	<code>TblHandle</code>	Handle of Table to be dumped.
----	------------------------	-------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.
<code>CFE_TBL_ERR_NO_ACCESS</code>	No Access.
<code>CFE_TBL_ERR_INVALID_HANDLE</code>	Invalid Handle.
<code>CFE_TBL_INFO_DUMP_PENDING</code>	Dump Pending.

See also

[CFE_TBL_Manage](#)

10.42.2.2 CFE_TBL_Load() `CFE_Status_t 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.

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	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be loaded.
in	<i>SrcType</i>	Flag indicating the nature of the given <i>SrcDataPtr</i> below. This value can be any one of the following: <ul style="list-style-type: none"> • CFE_TBL_SRC_FILE - File source When this option is selected, the <i>SrcDataPtr</i> 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 - Address source When this option is selected, the <i>SrcDataPtr</i> 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	<i>SrcDataPtr</i>	Pointer (must not be null) 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.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.

Return values

<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.
<i>CFE_TBL_ERR_DUMP_ONLY</i>	Dump Only Error.
<i>CFE_TBL_ERR_ILLEGAL_SRC_TYPE</i>	Illegal Source Type.
<i>CFE_TBL_ERR_LOAD_IN_PROGRESS</i>	Load In Progress.
<i>CFE_TBL_ERR_LOAD_INCOMPLETE</i>	Load Incomplete.
<i>CFE_TBL_ERR_NO_BUFFER_AVAIL</i>	No Buffer Available.
<i>CFE_TBL_ERR_ACCESS</i>	
<i>CFE_TBL_ERR_FILE_TOO_LARGE</i>	File Too Large.
<i>CFE_TBL_ERR_BAD_CONTENT_ID</i>	Bad Content ID.
<i>CFE_TBL_ERR_BAD_SUBTYPE_ID</i>	Bad Subtype ID.
<i>CFE_TBL_ERR_NO_STD_HEADER</i>	No Standard Header.
<i>CFE_TBL_ERR_NO_TBL_HEADER</i>	No Table Header.
<i>CFE_TBL_ERR_PARTIAL_LOAD</i>	Partial Load Error.
<i>CFE_TBL_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_TBL_Update](#), [CFE_TBL_Validate](#), [CFE_TBL_Manage](#)

Referenced by CF_TableInit().

10.42.2.3 CFE_TBL_Manage() *CFE_Status_t* CFE_TBL_Manage (
CFE_TBL_Handle_t TblHandle)

Perform standard operations to maintain a table.

Description

Applications should call this API periodically to process pending requests for update, validation, or dump to buffer. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be managed.
----	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.

Return values

<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.
<i>CFE_TBL_INFO_DUMP_PENDING</i>	Dump Pending.
<i>CFE_TBL_INFO_UPDATE_PENDING</i>	Update Pending.
<i>CFE_TBL_INFO_VALIDATION_PENDING</i>	

See also

[CFE_TBL_Update](#), [CFE_TBL_Validate](#), [CFE_TBL_Load](#), [CFE_TBL_DumpToBuffer](#)

Referenced by `CF_CheckTables()`, and `CF_TableInit()`.

10.42.2.4 CFE_TBL_Modified() [*CFE_Status_t*](#) `CFE_TBL_Modified (`
`CFE_TBL_Handle_t TblHandle)`

Notify cFE Table Services that table contents have been modified by the Application.

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

<code>in</code>	<code>TblHandle</code>	Handle of Table that was modified.
-----------------	------------------------	------------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

See also

[CFE_TBL_Manage](#)

10.42.2.5 CFE_TBL_Update() `CFE_Status_t CFE_TBL_Update (`
`CFE_TBL_Handle_t TblHandle)`

Update contents of a specified table, if an update is pending.

Description

Typically, apps should just call [CFE_TBL_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just an update should be performed.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be updated.
----	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_NO_UPDATE_PENDING	No Update Pending.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

[CFE_TBL_Load](#), [CFE_TBL_Validate](#), [CFE_TBL_Manage](#)

10.42.2.6 CFE_TBL_Validate() `CFE_Status_t CFE_TBL_Validate (`
`CFE_TBL_Handle_t TblHandle)`

Perform steps to validate the contents of a table image.

Description

Typically, apps should just call [CFE_TBL_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a validation should be performed.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be managed.
----	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_NO_VALIDATION_PENDING	
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

[CFE_TBL_Update](#), [CFE_TBL_Manage](#), [CFE_TBL_Load](#)

10.43 cFE Access Table Content APIs

Functions

- **`CFE_Status_t CFE_TBL_GetAddress`** (`void **TblPtr, CFE_TBL_Handle_t TblHandle`)

Obtain the current address of the contents of the specified table.
- **`CFE_Status_t CFE_TBL_ReleaseAddress`** (`CFE_TBL_Handle_t TblHandle`)

Release previously obtained pointer to the contents of the specified table.
- **`CFE_Status_t CFE_TBL_GetAddresses`** (`void **TblPtrs[], uint16 NumTables, const CFE_TBL_Handle_t TblHandles[]`)

Obtain the current addresses of an array of specified tables.
- **`CFE_Status_t CFE_TBL_ReleaseAddresses`** (`uint16 NumTables, const CFE_TBL_Handle_t TblHandles[]`)

Release the addresses of an array of specified tables.

10.43.1 Detailed Description

10.43.2 Function Documentation

10.43.2.1 `CFE_TBL_GetAddress()` `CFE_Status_t CFE_TBL_GetAddress (`
`void ** TblPtr,`
`CFE_TBL_Handle_t TblHandle)`

Obtain the current address of the contents of the specified table.

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.
3. `CFE_TBL_ERR_NEVER_LOADED` will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the `CFE_TBL_ReleaseAddress` API before the table can be loaded with data.

Parameters

<code>out</code>	<code>TblPtr</code>	The address of a pointer (must not be null) 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. <code>*TblPtr</code> is the address of the first byte of data associated with the specified table.
<code>in</code>	<code>TblHandle</code>	Handle, previously obtained from <code>CFE_TBL_Register</code> or <code>CFE_TBL_Share</code> , that identifies the Table whose address is to be returned.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_UPDATED	Updated.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.
CFE_TBL_ERR_UNREGISTERED	Unregistered.
CFE_TBL_ERR_NEVER_LOADED	Never Loaded.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

[CFE_TBL_ReleaseAddress](#), [CFE_TBL_GetAddresses](#), [CFE_TBL_ReleaseAddresses](#)

Referenced by CF_CheckTables(), and CF_TableInit().

10.43.2.2 CFE_TBL_GetAddresses() [CFE_Status_t](#) CFE_TBL_GetAddresses (

```
    void ** TblPtrs[],
    uint16 NumTables,
    const CFE_TBL_Handle_t TblHandles[] )
```

Obtain the current addresses of an array of specified tables.

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_GetAddress](#).

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.
3. [CFE_TBL_ERR_NEVER_LOADED](#) will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the [CFE_TBL_ReleaseAddress](#) API before the table can be loaded with data.

Parameters

<i>out</i>	<i>TblPtrs</i>	Array of Pointers (must not be null) to variables that calling Application wishes to hold the start addresses of the Tables. *TblPtrs is an array of addresses of the first byte of data associated with the specified tables.
<i>in</i>	<i>NumTables</i>	Size of TblPtrs and TblHandles arrays.
<i>in</i>	<i>TblHandles</i>	Array of Table Handles, previously obtained from CFE_TBL_Register or CFE_TBL_Share , of those tables whose start addresses are to be obtained.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_UPDATED	Updated.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.
CFE_TBL_ERR_UNREGISTERED	Unregistered.
CFE_TBL_ERR_NEVER_LOADED	Never Loaded.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

[CFE_TBL_GetAddress](#), [CFE_TBL_ReleaseAddress](#), [CFE_TBL_ReleaseAddresses](#)

10.43.2.3 CFE_TBL_ReleaseAddress()

```
CFE_Status_t CFE_TBL_ReleaseAddress (
    CFE_TBL_Handle_t TblHandle )
```

Release previously obtained pointer to the contents of the specified table.

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

<i>in</i>	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table whose address is to be released.
-----------	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.
<i>CFE_TBL_ERR_NEVER_LOADED</i>	Never Loaded.

See also

[CFE_TBL_GetAddress](#), [CFE_TBL_GetAddresses](#), [CFE_TBL_ReleaseAddresses](#)

Referenced by `CF_CheckTables()`.

```
10.43.2.4 CFE_TBL_ReleaseAddresses() CFE_Status_t CFE_TBL_ReleaseAddresses (
    uint16 NumTables,
    const CFE_TBL_Handle_t TblHandles[] )
```

Release the addresses of an array of specified tables.

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

<i>in</i>	<i>NumTables</i>	Size of TblHandles array.
<i>in</i>	<i>TblHandles</i>	Array of Table Handles (must not be null), previously obtained from CFE_TBL_Register or CFE_TBL_Share , of those tables whose start addresses are to be released.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

Return values

<i>CFE_TBL_ERR_NEVER_LOADED</i>	Never Loaded.
<i>CFE_TBL_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_TBL_GetAddress](#), [CFE_TBL_ReleaseAddress](#), [CFE_TBL_GetAddresses](#)

10.44 cFE Get Table Information APIs

Functions

- [CFE_Status_t CFE_TBL_GetStatus \(CFE_TBL_Handle_t TblHandle\)](#)
Obtain current status of pending actions for a table.
- [CFE_Status_t CFE_TBL_GetInfo \(CFE_TBL_Info_t *TblInfoPtr, const char *TblName\)](#)
Obtain characteristics/information of/about a specified table.
- [CFE_Status_t CFE_TBL_NotifyByMessage \(CFE_TBL_Handle_t TblHandle, CFE_SB_MsgId_t MsgId, CFE_MSG_FcnCode_t CommandCode, uint32 Parameter\)](#)
Instruct cFE Table Services to notify Application via message when table requires management.

10.44.1 Detailed Description

10.44.2 Function Documentation

10.44.2.1 CFE_TBL_GetInfo() [CFE_Status_t CFE_TBL_GetInfo \(](#)
[CFE_TBL_Info_t * TblInfoPtr,](#)
[const char * TblName \)](#)

Obtain characteristics/information of/about a specified table.

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

out	<i>TblInfoPtr</i>	A pointer to a CFE_TBL_Info_t data structure (must not be null) that is to be populated with table characteristics and information. *TblInfoPtr is the description of the tables characteristics and registry information stored in the CFE_TBL_Info_t data structure format.
in	<i>TblName</i>	The application specific name (must not be null) of the table of the form "AppName.RawTableName", where RawTableName is the name specified in the CFE_TBL_Register API call. Example: "ACS.TamParams" for a table called "TamParams" that was registered by the application called "ACS".

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_ERR_INVALID_NAME	Invalid Name.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also[CFE_TBL_GetStatus](#)**10.44.2.2 CFE_TBL_GetStatus()** `CFE_Status_t CFE_TBL_GetStatus (``CFE_TBL_Handle_t TblHandle)`

Obtain current status of pending actions for a table.

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	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , that identifies the Table to be managed.
----	------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_UPDATE_PENDING	Update Pending.
CFE_TBL_INFO_VALIDATION_PENDING	
CFE_TBL_INFO_DUMP_PENDING	Dump Pending.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

Note

Some status return codes are "success" while being non-zero. This behavior will change in the future.

See also[CFE_TBL_Manage](#), [CFE_TBL_Update](#), [CFE_TBL_Validate](#), [CFE_TBL_GetInfo](#)**10.44.2.3 CFE_TBL_NotifyByMessage()** `CFE_Status_t CFE_TBL_NotifyByMessage (``CFE_TBL_Handle_t TblHandle,``CFE_SB_MsgId_t MsgId,`

```
CFE_MSG_FcnCode_t CommandCode,  
    uint32 Parameter )
```

Instruct cFE Table Services to notify Application via message when table requires management.

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
- Recommend **NOT** using the ground command MID which typically impacts command counters. The typical approach is to use a unique MID for inter-task communications similar to how schedulers typically trigger application housekeeping messages.

Parameters

in	<i>TblHandle</i>	Handle of Table with which the message should be associated.
in	<i>MsgId</i>	Message ID to be used in notification message sent by Table Services.
in	<i>CommandCode</i>	Command Code value to be placed in secondary header of message sent by Table Services.
in	<i>Parameter</i>	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

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

[CFE_TBL_Register](#)

10.45 cFE Table Type Defines

Macros

- `#define CFE_TBL_OPT_BUFFER_MSK (0x0001)`
Table buffer mask.
- `#define CFE_TBL_OPT_SNGL_BUFFER (0x0000)`
Single buffer table.
- `#define CFE_TBL_OPT_DBL_BUFFER (0x0001)`
Double buffer table.
- `#define CFE_TBL_OPT_LD_DMP_MSK (0x0002)`
Table load/dump mask.
- `#define CFE_TBL_OPT_LOAD_DUMP (0x0000)`
Load/Dump table.
- `#define CFE_TBL_OPT_DUMP_ONLY (0x0002)`
Dump only table.
- `#define CFE_TBL_OPT_USR_DEF_MSK (0x0004)`
Table user defined mask.
- `#define CFE_TBL_OPT_NOT_USR_DEF (0x0000)`
Not user defined table.
- `#define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)`
User Defined table,..
- `#define CFE_TBL_OPT_CRITICAL_MSK (0x0008)`
Table critical mask.
- `#define CFE_TBL_OPT_NOT_CRITICAL (0x0000)`
Not critical table.
- `#define CFE_TBL_OPT_CRITICAL (0x0008)`
Critical table.
- `#define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)`
Default table options.

10.45.1 Detailed Description

10.45.2 Macro Definition Documentation

10.45.2.1 CFE_TBL_OPT_BUFFER_MSK `#define CFE_TBL_OPT_BUFFER_MSK (0x0001)`

Table buffer mask.

Definition at line 48 of file cfe_tbl_api_typedefs.h.

10.45.2.2 CFE_TBL_OPT_CRITICAL `#define CFE_TBL_OPT_CRITICAL (0x0008)`

Critical table.

Definition at line 63 of file cfe_tbl_api_typedefs.h.

10.45.2.3 CFE_TBL_OPT_CRITICAL_MSK `#define CFE_TBL_OPT_CRITICAL_MSK (0x0008)`

Table critical mask.

Definition at line 61 of file cfe_tbl_api_typedefs.h.

10.45.2.4 CFE_TBL_OPT_DBL_BUFFER #define CFE_TBL_OPT_DBL_BUFFER (0x0001)

Double buffer table.

Definition at line 50 of file cfe_tbl_api_typedefs.h.

10.45.2.5 CFE_TBL_OPT_DEFAULT #define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)

Default table options.

Definition at line 66 of file cfe_tbl_api_typedefs.h.

10.45.2.6 CFE_TBL_OPT_DUMP_ONLY #define CFE_TBL_OPT_DUMP_ONLY (0x0002)

Dump only table.

Definition at line 54 of file cfe_tbl_api_typedefs.h.

10.45.2.7 CFE_TBL_OPT_LD_DMP_MSK #define CFE_TBL_OPT_LD_DMP_MSK (0x0002)

Table load/dump mask.

Definition at line 52 of file cfe_tbl_api_typedefs.h.

10.45.2.8 CFE_TBL_OPT_LOAD_DUMP #define CFE_TBL_OPT_LOAD_DUMP (0x0000)

Load/Dump table.

Definition at line 53 of file cfe_tbl_api_typedefs.h.

10.45.2.9 CFE_TBL_OPT_NOT_CRITICAL #define CFE_TBL_OPT_NOT_CRITICAL (0x0000)

Not critical table.

Definition at line 62 of file cfe_tbl_api_typedefs.h.

10.45.2.10 CFE_TBL_OPT_NOT_USR_DEF #define CFE_TBL_OPT_NOT_USR_DEF (0x0000)

Not user defined table.

Definition at line 57 of file cfe_tbl_api_typedefs.h.

10.45.2.11 CFE_TBL_OPT_SNGL_BUFFER #define CFE_TBL_OPT_SNGL_BUFFER (0x0000)

Single buffer table.

Definition at line 49 of file cfe_tbl_api_typedefs.h.

10.45.2.12 CFE_TBL_OPT_USR_DEF_ADDR #define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)

User Defined table.,.

Note

Automatically includes [CFE_TBL_OPT_DUMP_ONLY](#) option

Definition at line 58 of file cfe_tbl_api_typedefs.h.

10.45.2.13 CFE_TBL_OPT_USR_DEF_MSK #define CFE_TBL_OPT_USR_DEF_MSK (0x0004)

Table user defined mask.

Definition at line 56 of file cfe_tbl_api_typedefs.h.

10.46 cFE Get Current Time APIs

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 (MET + SCTF) time.
- [CFE_TIME_SysTime_t CFE_TIME_GetUTC \(void\)](#)
Get the current UTC (MET + SCTF - Leap Seconds) time.
- [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.

10.46.1 Detailed Description

10.46.2 Function Documentation

10.46.2.1 CFE_TIME_GetMET() [CFE_TIME_SysTime_t CFE_TIME_GetMET \(void\)](#)

Get the current value of the Mission Elapsed Time (MET).

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

Returns

The current MET

See also

[CFE_TIME_GetTime](#), [CFE_TIME_GetTAI](#), [CFE_TIME_GetUTC](#), [CFE_TIME_GetMETseconds](#), [CFE_TIME_GetMETsubsecs](#), [CFE_TIME_MET2SCTime](#)

10.46.2.2 CFE_TIME_GetMETseconds() [uint32 CFE_TIME_GetMETseconds \(void\)](#)

Get the current seconds count of the mission-elapsed time.

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](#)

10.46.2.3 CFE_TIME_GetMETsubsecs() `uint32 CFE_TIME_GetMETsubsecs (`
 `void)`

Get the current sub-seconds count of the mission-elapsed time.

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](#)

10.46.2.4 CFE_TIME_GetTAI() `CFE_TIME_SysTime_t CFE_TIME_GetTAI (`
 `void)`

Get the current TAI (MET + SCTF) time.

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.

Returns

The current spacecraft time in TAI

See also

[CFE_TIME_GetTime](#), [CFE_TIME_GetUTC](#), [CFE_TIME_GetMET](#), [CFE_TIME_GetMETseconds](#), [CFE_TIME_GetMETsubsecs](#)

10.46.2.5 CFE_TIME_GetTime() `CFE_TIME_SysTime_t CFE_TIME_GetTime (void)`

Get the current spacecraft time.

Description

This routine returns the current spacecraft time, which is the amount of time elapsed since the epoch as set in mission configuration. 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 rather than the specific routines for getting UTC/TAI directly.

Assumptions, External Events, and Notes:

None

Returns

The current spacecraft time in default format

See also

[CFE_TIME_GetTAI](#), [CFE_TIME_GetUTC](#), [CFE_TIME_GetMET](#), [CFE_TIME_GetMETseconds](#), [CFE_TIME_GetMETsubsecs](#)

Referenced by CF_CFDP_Send().

10.46.2.6 CFE_TIME_GetUTC() `CFE_TIME_SysTime_t CFE_TIME_GetUTC (void)`

Get the current UTC (MET + SCTF - Leap Seconds) time.

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.

Returns

The current spacecraft time in UTC

See also

[CFE_TIME_GetTime](#), [CFE_TIME_GetTAI](#), [CFE_TIME_GetMET](#), [CFE_TIME_GetMETseconds](#), [CFE_TIME_GetMETsubsecs](#)

10.47 cFE Get Time Information APIs

Functions

- [CFE_TIME_SysTime_t CFE_TIME_GetSTCF \(void\)](#)
Get the current value of the spacecraft time correction factor (STCF).
- [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.

10.47.1 Detailed Description

10.47.2 Function Documentation

10.47.2.1 CFE_TIME_GetClockInfo() [uint16 CFE_TIME_GetClockInfo \(void\)](#)

Provides information about the spacecraft clock.

Description

This routine returns information on the spacecraft clock in a bit mask.

Assumptions, External Events, and Notes:

None

Returns

Spacecraft clock information, [cFE Clock State Flag Defines](#). To extract the information from the returned value, the flags can be used as in the following:

```
if ((ReturnValue & CFE_TIME_FLAG_xxxxxxx) == CFE_TIME_FLAG_xxxxxxx) then the following definition of the CFE_TIME_FLAG_xxxxxxx is true.
```

See also

[CFE_TIME_GetSTCF](#), [CFE_TIME_GetLeapSeconds](#), [CFE_TIME_GetClockState](#)

10.47.2.2 CFE_TIME_GetClockState() [CFE_TIME_ClockState_Enum_t CFE_TIME_GetClockState \(void\)](#)

Get the current state of the spacecraft clock.

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

Returns

The current spacecraft clock state

See also

[CFE_TIME_GetSTCF](#), [CFE_TIME_GetLeapSeconds](#), [CFE_TIME_GetClockInfo](#)

10.47.2.3 CFE_TIME_GetLeapSeconds() `int16 CFE_TIME_GetLeapSeconds (void)`

Get the current value of the leap seconds counter.

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). There is no API provided to set or adjust leap seconds or STCF, those actions should be done by command only. This API is provided for applications to be able to include leap seconds 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

Returns

The current spacecraft leap seconds.

See also

[CFE_TIME_GetSTCF](#), [CFE_TIME_GetClockState](#), [CFE_TIME_GetClockInfo](#)

10.47.2.4 CFE_TIME_GetSTCF() `CFE_TIME_SysTime_t CFE_TIME_GetSTCF (void)`

Get the current value of the spacecraft time correction factor (STCF).

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. There is no API provided to set or adjust leap seconds or STCF, those actions should be done by command only. This API is provided for applications to be able to include STCF in their data products to aid in time correlation during downstream science data processing.

Assumptions, External Events, and Notes:

Does not include leap seconds

Returns

The current SCTF

See also

[CFE_TIME_GetLeapSeconds](#), [CFE_TIME_GetClockState](#), [CFE_TIME_GetClockInfo](#)

10.48 cFE Time Arithmetic APIs

Functions

- `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.

10.48.1 Detailed Description

10.48.2 Function Documentation

10.48.2.1 CFE_TIME_Add() `CFE_TIME_SysTime_t CFE_TIME_Add (`
 `CFE_TIME_SysTime_t Time1,`
 `CFE_TIME_SysTime_t Time2)`

Adds two time values.

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	<i>Time1</i>	The first time to be added.
in	<i>Time2</i>	The second time to be added.

Returns

The sum of the two times. 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).

See also

[CFE_TIME_Subtract](#), [CFE_TIME_Compare](#)

10.48.2.2 CFE_TIME_Compare() `CFE_TIME_Compare_t CFE_TIME_Compare (`
 `CFE_TIME_SysTime_t TimeA,`
 `CFE_TIME_SysTime_t TimeB)`

Compares two time values.

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	<i>TimeA</i>	The first time to compare.
in	<i>TimeB</i>	The second time to compare.

Returns

The result of comparing the two times.

Return values

<i>CFE_TIME_EQUAL</i>	The two specified times are considered to be equal.
<i>CFE_TIME_A_GT_B</i>	The first specified time is considered to be after the second specified time.
<i>CFE_TIME_A_LT_B</i>	The first specified time is considered to be before the second specified time.

See also

[CFE_TIME_Add](#), [CFE_TIME_Subtract](#)

10.48.2.3 CFE_TIME_Subtract()

```
CFE_TIME_SysTime_t CFE_TIME_Subtract (
    CFE_TIME_SysTime_t Time1,
    CFE_TIME_SysTime_t Time2 )
```

Subtracts two time values.

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	<i>Time1</i>	The base time.
in	<i>Time2</i>	The time to be subtracted from the base time.

Returns

The result of subtracting the two times. If the subtraction results in an underflow, the result will roll over (this is not considered an error).

See also

[CFE_TIME_Add](#), [CFE_TIME_Compare](#)

10.49 cFE Time Conversion APIs

Functions

- `CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (CFE_TIME_SysTime_t METTime)`
Convert specified MET into Spacecraft Time.
- `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.

10.49.1 Detailed Description

10.49.2 Function Documentation

10.49.2.1 CFE_TIME_MET2SCTime() `CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (` `CFE_TIME_SysTime_t METTime)`

Convert specified MET into Spacecraft Time.

Description

This function returns Spacecraft Time given MET. Note that Spacecraft Time is returned as either UTC or TAI depending 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	<code>METTime</code>	The MET to be converted.
----	----------------------	--------------------------

Returns

Spacecraft Time (UTC or TAI) corresponding to the specified MET

See also

`CFE_TIME_GetMET`, `CFE_TIME_GetMETseconds`, `CFE_TIME_GetMETsubsecs`, `CFE_TIME_Sub2MicroSecs`, `CFE_TIME_Micro2SubSecs`

10.49.2.2 CFE_TIME_Micro2SubSecs() `uint32 CFE_TIME_Micro2SubSecs (` `uint32 MicroSeconds)`

Converts a number of microseconds to an equivalent sub-seconds count.

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	<i>MicroSeconds</i>	The sub-seconds count to convert.
----	---------------------	-----------------------------------

Returns

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.

See also

[CFE_TIME_MET2SCTime](#), [CFE_TIME_Sub2MicroSecs](#),

10.49.2.3 CFE_TIME_Sub2MicroSecs() `uint32 CFE_TIME_Sub2MicroSecs (`
`uint32 SubSeconds)`

Converts a sub-seconds count to an equivalent number of microseconds.

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	<i>SubSeconds</i>	The sub-seconds count to convert.
----	-------------------	-----------------------------------

Returns

The equivalent number of microseconds.

See also

[CFE_TIME_MET2SCTime](#), [CFE_TIME_Micro2SubSecs](#),

10.50 cFE External Time Source APIs

Functions

- 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.
- [CFE_Status_t CFE_TIME_RegisterSynchCallback](#) ([CFE_TIME_SynchCallbackPtr_t](#) CallbackFuncPtr)

Registers a callback function that is called whenever time synchronization occurs.
- [CFE_Status_t CFE_TIME_UnregisterSynchCallback](#) ([CFE_TIME_SynchCallbackPtr_t](#) CallbackFuncPtr)

Unregisters a callback function that is called whenever time synchronization occurs.

10.50.1 Detailed Description

10.50.2 Function Documentation

10.50.2.1 [CFE_TIME_ExternalGPS\(\)](#) `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.

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	<i>NewTime</i>	The MET value at the next (or previous) 1 Hz tone signal.
in	<i>NewLeaps</i>	The Leap Seconds value used to calculate time as UTC.

See also

[CFE_TIME_ExternalTone](#), [CFE_TIME_ExternalMET](#), [CFE_TIME_ExternalTime](#)

10.50.2.2 CFE_TIME_ExternalMET() `void CFE_TIME_ExternalMET (`
`CFE_TIME_SysTime_t NewMET)`

Provides the Mission Elapsed Time from an external source.

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	<i>NewMET</i>	The MET value at the next (or previous) 1 Hz tone signal.
----	---------------	---

See also

[CFE_TIME_ExternalTone](#), [CFE_TIME_ExternalGPS](#), [CFE_TIME_ExternalTime](#)

10.50.2.3 CFE_TIME_ExternalTime() `void CFE_TIME_ExternalTime (`
`CFE_TIME_SysTime_t NewTime)`

Provide the time from an external source that measures time relative to a known epoch.

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 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_TIME`, which indicates that the external time data consists of a time value relative to a known epoch.

Parameters

in	<code>NewTime</code>	The MET value at the next (or previous) 1 Hz tone signal.
----	----------------------	---

See also

[CFE_TIME_ExternalTone](#), [CFE_TIME_ExternalMET](#), [CFE_TIME_ExternalGPS](#)

10.50.2.4 CFE_TIME_ExternalTone() `void CFE_TIME_ExternalTone (`
`void)`

Provides the 1 Hz signal from an external source.

Description

This routine provides a method for cFE TIME software to be notified of the occurrence 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](#)

10.50.2.5 CFE_TIME_RegisterSynchCallback() `CFE_Status_t CFE_TIME_RegisterSynchCallback (`
`CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)`

Registers a callback function that is called whenever time synchronization occurs.

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.

Parameters

in	<i>CallbackFuncPtr</i>	Function to call at synchronization interval (must not be null)
----	------------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TIME_TOO_MANY_SYNCH_CALLBACKS</i>	Too Many Sync Callbacks.
<i>CFE_TIME_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_TIME_UnregisterSynchCallback](#)

10.50.2.6 CFE_TIME_UnregisterSynchCallback() *CFE_Status_t CFE_TIME_UnregisterSynchCallback (CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)*

Unregisters a callback function that is called whenever time synchronization occurs.

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).

Parameters

in	<i>CallbackFuncPtr</i>	Function to remove from synchronization call list (must not be null)
----	------------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TIME_CALLBACK_NOT_REGISTERED</i>	Callback Not Registered.
<i>CFE_TIME_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_TIME_RegisterSyncCallback](#)

10.51 cFE Miscellaneous Time APIs

Functions

- void `CFE_TIME_Print` (char *PrintBuffer, `CFE_TIME_SysTime_t` TimeToPrint)

Print a time value as a string.

- void `CFE_TIME_Local1HzISR` (void)

This function is called via a timer callback set up at initialization of the TIME service.

10.51.1 Detailed Description

10.51.2 Function Documentation

10.51.2.1 `CFE_TIME_Local1HzISR()` void CFE_TIME_Local1HzISR (

`void`)

This function is called via a timer callback set up at initialization of the TIME service.

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.

10.51.2.2 `CFE_TIME_Print()` void CFE_TIME_Print (

`char * PrintBuffer,`

`CFE_TIME_SysTime_t TimeToPrint`)

Print a time value as a string.

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:

- The value of the time argument is simply added to the configuration definitions for the ground epoch and converted into a fixed length string in the buffer provided by the caller.
- A loss of data during the string conversion will occur if the computed year exceeds 9999. However, a year that large would require an unrealistic definition for the ground epoch since the maximum amount of time represented by a [CFE_TIME_SysTime](#) structure is approximately 136 years.

Parameters

out	<i>PrintBuffer</i>	Pointer to a character array (must not be null) of at least CFE_TIME_PRINTED_STRING_SIZE characters in length. *PrintBuffer is the time as a character string as described above.
in	<i>TimeToPrint</i>	The time to print into the character array.

10.52 cFE Resource ID base values

Enumerations

- enum {

`CFE_RESOURCEID_ES_TASKID_BASE_OFFSET = OS_OBJECT_TYPE_OS_TASK, CFE_RESOURCEID_ES_APPID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 1, CFE_RESOURCEID_ES_LIBID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 2, CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 3, CFE_RESOURCEID_ES_POOLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 4, CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 5, CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET = OS_OBJECT_TYPE_USER + 6, CFE_RESOURCEID_CONFIGID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 7,`

`CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 8, CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 9 }`
- enum {

`CFE_ES_TASKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_OFFSET), CFE_ES_APPID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BASE_OFFSET), CFE_ES_LIBID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_LIBID_BASE_OFFSET), CFE_ES_COUNTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET), CFE_ES_POOLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_OFFSET), CFE_ES_CDSBLOCKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET), CFE_SB_PIPEID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET), CFE_CONFIGID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_CONFIGID_BASE_OFFSET), CFE_TBL_VALRESULTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET), CFE_TBL_DUMPCTRLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET) }`

10.52.1 Detailed Description

10.52.2 Enumeration Type Documentation

10.52.2.1 anonymous enum anonymous enum

Enumerator

CFE_RESOURCEID_ES_TASKID_BASE_OFFSET
CFE_RESOURCEID_ES_APPID_BASE_OFFSET
CFE_RESOURCEID_ES_LIBID_BASE_OFFSET
CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET
CFE_RESOURCEID_ES_POOLID_BASE_OFFSET
CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET
CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET
CFE_RESOURCEID_CONFIGID_BASE_OFFSET
CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET
CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET

Definition at line 48 of file cfe_core_resourceid_basevalues.h.

10.52.2.2 anonymous enum anonymous enum

Enumerator

CFE_ES_TASKID_BASE	
CFE_ES_APPID_BASE	
CFE_ES_LIBID_BASE	
CFE_ES_COUNTID_BASE	
CFE_ES_POOLID_BASE	
CFE_ES_CDSBLOCKID_BASE	
CFE_SB_PIPEID_BASE	
CFE_CONFIGID_BASE	
CFE_TBL_VALRESULTID_BASE	
CFE_TBL_DUMPCTRLID_BASE	

Definition at line 85 of file `cfe_core_resourceid_basevalues.h`.

10.53 cFE Clock State Flag Defines

Macros

- #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_REFERR 0x0010
GetReference read error, will be set if unable to get a consistent ref value.
- #define CFE_TIME_FLAG_UNUSED 0x000F
Reserved flags - should be zero.

10.53.1 Detailed Description

10.53.2 Macro Definition Documentation

10.53.2.1 CFE_TIME_FLAG_ADD1HZ #define CFE_TIME_FLAG_ADD1HZ 0x0100

1 Hz STCF Adjustment is to be done in a positive direction

Definition at line 44 of file default_cfe_time_msgdefs.h.

10.53.2.2 CFE_TIME_FLAG_ADDADJ #define CFE_TIME_FLAG_ADDADJ 0x0200

One time STCF Adjustment is to be done in positive direction.

Definition at line 43 of file default_cfe_time_msgdefs.h.

10.53.2.3 CFE_TIME_FLAG_ADDTCL #define CFE_TIME_FLAG_ADDTCL 0x0080

Time Client Latency is applied in a positive direction.

Definition at line 45 of file default_cfe_time_msgdefs.h.

10.53.2.4 CFE_TIME_FLAG_CLKSET #define CFE_TIME_FLAG_CLKSET 0x8000

The spacecraft time has been set.

Definition at line 37 of file default_cfe_time_msgdefs.h.

10.53.2.5 CFE_TIME_FLAG_CMDFLY #define CFE_TIME_FLAG_CMDFLY 0x0400

This instance of Time Services was commanded into flywheel mode.

Definition at line 42 of file default_cfe_time_msgdefs.h.

10.53.2.6 CFE_TIME_FLAG_FLYING #define CFE_TIME_FLAG_FLYING 0x4000

This instance of Time Services is flywheeling.

Definition at line 38 of file default_cfe_time_msgdefs.h.

10.53.2.7 CFE_TIME_FLAG_GDTONE #define CFE_TIME_FLAG_GDTONE 0x0020

The tone received is good compared to the last tone received.

Definition at line 47 of file default_cfe_time_msgdefs.h.

10.53.2.8 CFE_TIME_FLAG_REFERR #define CFE_TIME_FLAG_REFERR 0x0010

GetReference read error, will be set if unable to get a consistent ref value.

Definition at line 48 of file default_cfe_time_msgdefs.h.

10.53.2.9 CFE_TIME_FLAG_SERVER #define CFE_TIME_FLAG_SERVER 0x0040

This instance of Time Services is a Time Server.

Definition at line 46 of file default_cfe_time_msgdefs.h.

10.53.2.10 CFE_TIME_FLAG_SIGPRI #define CFE_TIME_FLAG_SIGPRI 0x1000

The clock signal is set to "primary".

Definition at line 40 of file default_cfe_time_msgdefs.h.

10.53.2.11 CFE_TIME_FLAG_SRCINT #define CFE_TIME_FLAG_SRCINT 0x2000

The clock source is set to "internal".

Definition at line 39 of file default_cfe_time_msgdefs.h.

10.53.2.12 CFE_TIME_FLAG_SRVFLY #define CFE_TIME_FLAG_SRVFLY 0x0800

The Time Server is in flywheel mode.

Definition at line 41 of file default_cfe_time_msgdefs.h.

10.53.2.13 CFE_TIME_FLAG_UNUSED #define CFE_TIME_FLAG_UNUSED 0x000F

Reserved flags - should be zero.

Definition at line 50 of file default_cfe_time_msgdefs.h.

10.54 OSAL Semaphore State Defines

Macros

- `#define OS_SEM_FULL 1`
Semaphore full state.
- `#define OS_SEM_EMPTY 0`
Semaphore empty state.

10.54.1 Detailed Description

10.54.2 Macro Definition Documentation

10.54.2.1 OS_SEM_EMPTY `#define OS_SEM_EMPTY 0`

Semaphore empty state.

Definition at line 35 of file osapi-binsem.h.

10.54.2.2 OS_SEM_FULL `#define OS_SEM_FULL 1`

Semaphore full state.

Definition at line 34 of file osapi-binsem.h.

10.55 OSAL Binary Semaphore APIs

Functions

- `int32 OS_BinSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)`
Creates a binary semaphore.
- `int32 OS_BinSemFlush (osal_id_t sem_id)`
Unblock all tasks pending on the specified semaphore.
- `int32 OS_BinSemGive (osal_id_t sem_id)`
Increment the semaphore value.
- `int32 OS_BinSemTake (osal_id_t sem_id)`
Decrement the semaphore value.
- `int32 OS_BinSemTimedWait (osal_id_t sem_id, uint32 msecs)`
Decrement the semaphore value with a timeout.
- `int32 OS_BinSemDelete (osal_id_t sem_id)`
Deletes the specified Binary Semaphore.
- `int32 OS_BinSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`
Find an existing semaphore ID by name.
- `int32 OS_BinSemGetInfo (osal_id_t sem_id, OS_bin_sem_prop_t *bin_prop)`
Fill a property object buffer with details regarding the resource.

10.55.1 Detailed Description

10.55.2 Function Documentation

10.55.2.1 OS_BinSemCreate() `int32 OS_BinSemCreate (`
 `osal_id_t * sem_id,`
 `const char * sem_name,`
 `uint32 sem_initial_value,`
 `uint32 options)`

Creates a binary semaphore.

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

<code>out</code>	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>sem_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>sem_initial_value</code>	the initial value of the binary semaphore
<code>in</code>	<code>options</code>	Reserved for future use, should be passed as 0.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if sem name or sem_id are NULL

Return values

<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if all of the semaphore ids are taken
<i>OS_ERR_NAME_TAKEN</i>	if this is already the name of a binary semaphore
<i>OS_SEM_FAILURE</i>	if the OS call failed (return value only verified in coverage test)

```
10.55.2.2 OS_BinSemDelete() int32 OS_BinSemDelete (
    osal_id_t sem_id )
```

Deletes the specified Binary Semaphore.

This is the function used to delete a binary semaphore in the operating system. This also frees the respective *sem_id* to be used again when another semaphore is created.

Parameters

in	<i>sem_id</i>	The object ID to delete
----	---------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid binary semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified failure occurs (return value only verified in coverage test)

```
10.55.2.3 OS_BinSemFlush() int32 OS_BinSemFlush (
    osal_id_t sem_id )
```

Unblock all tasks pending on the specified semaphore.

The function unblocks all tasks pending on the specified semaphore. However, this function does not change the state of the semaphore.

Parameters

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
-------------------	-----------------------

Return values

<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a binary semaphore
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

```
10.55.2.4 OS_BinSemGetIdByName() int32 OS_BinSemGetIdByName (
    osal_id_t * sem_id,
    const char * sem_name )
```

Find an existing semaphore ID by 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	<code>sem_id</code>	will be set to the ID of the existing resource
in	<code>sem_name</code>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

```
10.55.2.5 OS_BinSemGetInfo() int32 OS_BinSemGetInfo (
    osal_id_t sem_id,
    OS_bin_sem_prop_t * bin_prop )
```

Fill a property object buffer with details regarding the resource.

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	<code>sem_id</code>	The object ID to operate on
out	<code>bin_prop</code>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
-------------------------	-----------------------

Return values

<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the bin_prop pointer is null
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.

10.55.2.6 OS_BinSemGive() `int32 OS_BinSemGive (osal_id_t sem_id)`

Increment the semaphore value.

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	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a binary semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified failure occurs (return value only verified in coverage test)

10.55.2.7 OS_BinSemTake() `int32 OS_BinSemTake (osal_id_t sem_id)`

Decrement the semaphore value.

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	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
-------------------	-----------------------

Return values

OS_ERR_INVALID_ID	the Id passed in is not a valid binary semaphore
OS_SEM_FAILURE	if an unspecified failure occurs (return value only verified in coverage test)

```
10.55.2.8 OS_BinSemTimedWait() int32 OS_BinSemTimedWait (
    osal_id_t sem_id,
    uint32 msecs )
```

Decrement the semaphore value with a timeout.

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	<code>sem_id</code>	The object ID to operate on
in	<code>msecs</code>	The maximum amount of time to block, in milliseconds

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_SEM_TIMEOUT	if semaphore was not relinquished in time
OS_ERR_INVALID_ID	if the ID passed in is not a valid semaphore ID
OS_SEM_FAILURE	if an unspecified failure occurs (return value only verified in coverage test)

10.56 OSAL BSP low level access APIs

These are for OSAL internal BSP information access to pass any BSP-specific boot/command line/startup arguments through to the application, and return a status code back to the OS after exit.

Functions

- void `OS_BSP_SetResourceTypeConfig (uint32 ResourceType, uint32 ConfigOptionValue)`
- `uint32 OS_BSP_GetResourceTypeConfig (uint32 ResourceType)`
- `uint32 OS_BSP_GetArgC (void)`
- `char *const * OS_BSP_GetArgV (void)`
- void `OS_BSP_SetExitCode (int32 code)`

10.56.1 Detailed Description

These are for OSAL internal BSP information access to pass any BSP-specific boot/command line/startup arguments through to the application, and return a status code back to the OS after exit.

Not intended for user application use

10.56.2 Function Documentation

10.56.2.1 OS_BSP_GetArgC() `uint32 OS_BSP_GetArgC (`
`void)`

10.56.2.2 OS_BSP_GetArgV() `char* const * OS_BSP_GetArgV (`
`void)`

10.56.2.3 OS_BSP_GetResourceTypeConfig() `uint32 OS_BSP_GetResourceTypeConfig (`
`uint32 ResourceType)`

10.56.2.4 OS_BSP_SetExitCode() `void OS_BSP_SetExitCode (`
`int32 code)`

10.56.2.5 OS_BSP_SetResourceTypeConfig() `void OS_BSP_SetResourceTypeConfig (`
`uint32 ResourceType,`
`uint32 ConfigOptionValue)`

10.57 OSAL Real Time Clock APIs

Functions

- `int32 OS_GetLocalTime (OS_time_t *time_struct)`
Get the local time.
- `int32 OS_SetLocalTime (const OS_time_t *time_struct)`
Set the local time.
- `OS_time_t OS_TimeFromRelativeMilliseconds (int32 relative_msec)`
Gets an absolute time value relative to the current time.
- `int32 OS_TimeToRelativeMilliseconds (OS_time_t time)`
Gets a relative time value from an absolute time.
- static `int64 OS_TimeGetTotalSeconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to whole number of seconds.
- static `OS_time_t OS_TimeFromTotalSeconds (int64 tm)`
Get an `OS_time_t` interval object from an integer number of seconds.
- static `int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to millisecond units.
- static `OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of milliseconds.
- static `int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to microsecond units.
- static `OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of microseconds.
- static `int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to nanosecond units.
- static `OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of nanoseconds.
- static `uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`
Get subseconds portion (fractional part only) from an `OS_time_t` object.
- static `uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`
Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.
- static `uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`
Get milliseconds portion (fractional part only) from an `OS_time_t` object.
- static `uint32 OS_TimeGetNanosecondsPart (OS_time_t tm)`
Get microseconds portion (fractional part only) from an `OS_time_t` object.
- static `OS_time_t OS_TimeAssembleFromNanoseconds (int64 seconds, uint32 nanoseconds)`
Assemble/Convert a number of seconds + nanoseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromMicroseconds (int64 seconds, uint32 microseconds)`
Assemble/Convert a number of seconds + microseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromMilliseconds (int64 seconds, uint32 milliseconds)`
Assemble/Convert a number of seconds + milliseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromSubseconds (int64 seconds, uint32 subseconds)`
Assemble/Convert a number of seconds + subseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAdd (OS_time_t time1, OS_time_t time2)`
Computes the sum of two time intervals.

- static `OS_time_t OS_TimeSubtract (OS_time_t time1, OS_time_t time2)`
Computes the difference between two time intervals.
- static bool `OS_TimeEqual (OS_time_t time1, OS_time_t time2)`
Checks if two time values are equal.
- static int8_t `OS_TimeGetSign (OS_time_t time)`
Checks the sign of the time value.
- static int8_t `OS_TimeCompare (OS_time_t time1, OS_time_t time2)`
Compares two time values.

10.57.1 Detailed Description

10.57.2 Function Documentation

10.57.2.1 `OS_GetLocalTime()` `int32 OS_GetLocalTime (OS_time_t * time_struct)`

Get the local time.

This function gets the local time from the underlying OS.

Note

Mission time management typically uses the cFE Time Service

Parameters

out	<code>time_struct</code>	An <code>OS_time_t</code> that will be set to the current time (must not be null)
-----	--------------------------	---

Returns

Get local time status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if time_struct is null

10.57.2.2 `OS_SetLocalTime()` `int32 OS_SetLocalTime (const OS_time_t * time_struct)`

Set the local time.

This function sets the local time on the underlying OS.

Note

Mission time management typically uses the cFE Time Services

Parameters

in	<code>time_struct</code>	An <code>OS_time_t</code> containing the current time (must not be null)
----	--------------------------	--

Returns

Set local time status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if time_struct is null

10.57.2.3 OS_TimeAdd() static `OS_time_t` OS_TimeAdd (
 `OS_time_t` time1,
 `OS_time_t` time2) [inline], [static]

Computes the sum of two time intervals.

Parameters

in	<i>time1</i>	The first interval
in	<i>time2</i>	The second interval

Returns

The sum of the two intervals (time1 + time2)

Definition at line 530 of file osapi-clock.h.

References `OS_time_t::ticks`.

10.57.2.4 OS_TimeAssembleFromMicroseconds() static `OS_time_t` OS_TimeAssembleFromMicroseconds (
 `int64` seconds,
 `uint32` microseconds) [inline], [static]

Assemble/Convert a number of seconds + microseconds into an `OS_time_t` interval.

This creates an `OS_time_t` value using a whole number of seconds and a fractional part in units of microseconds. This is the inverse of `OS_TimeGetTotalSeconds()` and `OS_TimeGetMicrosecondsPart()`, and should recreate the original `OS_time_t` value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

See also

[OS_TimeGetTotalSeconds\(\)](#), [OS_TimeGetMicrosecondsPart\(\)](#)

Parameters

in	<i>seconds</i>	Whole number of seconds
in	<i>microseconds</i>	Number of microseconds (fractional part only)

Returns

The input arguments represented as an `OS_time_t` interval

Definition at line 465 of file osapi-clock.h.

References `OS_TIME_TICKS_PER_SECOND`, `OS_TIME_TICKS_PER_USEC`, and `OS_time_t::ticks`.

```
10.57.2.5 OS_TimeAssembleFromMilliseconds() static OS\_time\_t OS_TimeAssembleFromMilliseconds (
    int64 seconds,
    uint32 milliseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + milliseconds into an [OS_time_t](#) interval.

This creates an [OS_time_t](#) value using a whole number of seconds and a fractional part in units of milliseconds. This is the inverse of [OS_TimeGetTotalSeconds\(\)](#) and [OS_TimeGetMillisecondsPart\(\)](#), and should recreate the original [OS_time_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

See also

[OS_TimeGetTotalSeconds\(\)](#), [OS_TimeGetMillisecondsPart\(\)](#)

Parameters

in	<i>seconds</i>	Whole number of seconds
in	<i>milliseconds</i>	Number of milliseconds (fractional part only)

Returns

The input arguments represented as an [OS_time_t](#) interval

Definition at line 489 of file osapi-clock.h.

References [OS_TIME_TICKS_PER_MSEC](#), [OS_TIME_TICKS_PER_SECOND](#), and [OS_time_t::ticks](#).

```
10.57.2.6 OS_TimeAssembleFromNanoseconds() static OS\_time\_t OS_TimeAssembleFromNanoseconds (
    int64 seconds,
    uint32 nanoseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + nanoseconds into an [OS_time_t](#) interval.

This creates an [OS_time_t](#) value using a whole number of seconds and a fractional part in units of nanoseconds. This is the inverse of [OS_TimeGetTotalSeconds\(\)](#) and [OS_TimeGetNanosecondsPart\(\)](#), and should recreate the original [OS_time_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

See also

[OS_TimeGetTotalSeconds\(\)](#), [OS_TimeGetNanosecondsPart\(\)](#)

Parameters

in	<i>seconds</i>	Whole number of seconds
in	<i>nanoseconds</i>	Number of nanoseconds (fractional part only)

Returns

The input arguments represented as an [OS_time_t](#) interval

Definition at line 441 of file osapi-clock.h.

References [OS_TIME_TICK_RESOLUTION_NS](#), [OS_TIME_TICKS_PER_SECOND](#), and [OS_time_t::ticks](#).

```
10.57.2.7 OS_TimeAssembleFromSubseconds() static OS\_time\_t OS_TimeAssembleFromSubseconds (
    int64 seconds,
```

```
    uint32 subseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + subseconds into an `OS_time_t` interval.

This creates an `OS_time_t` value using a whole number of seconds and a fractional part in units of sub-seconds ($1/2^{32}$). This is the inverse of `OS_TimeGetTotalSeconds()` and `OS_TimeGetSubsecondsPart()`, and should recreate the original `OS_time_t` value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

See also

[OS_TimeGetTotalSeconds\(\)](#), [OS_TimeGetNanosecondsPart\(\)](#)

Parameters

in	<i>seconds</i>	Whole number of seconds
in	<i>subseconds</i>	Number of subseconds (32 bit fixed point fractional part)

Returns

The input arguments represented as an `OS_time_t` interval

Definition at line 512 of file osapi-clock.h.

References `OS_TIME_TICKS_PER_SECOND`, and `OS_time_t::ticks`.

10.57.2.8 OS_TimeCompare() static int8_t OS_TimeCompare (
 OS_time_t time1,
 OS_time_t time2) [inline], [static]

Compares two time values.

Parameters

in	<i>time1</i>	The first time
in	<i>time2</i>	The second time

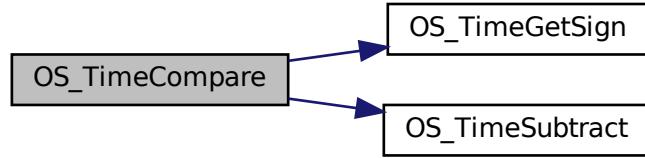
Return values

-1	if the <code>time1 < time2</code>
0	if the times are equal
1	if the <code>time1 > time2</code>

Definition at line 592 of file osapi-clock.h.

References `OS_TimeGetSign()`, and `OS_TimeSubtract()`.

Here is the call graph for this function:



10.57.2.9 OS_TimeEqual() `static bool OS_TimeEqual (`
 `OS_time_t time1,`
 `OS_time_t time2) [inline], [static]`

Checks if two time values are equal.

Parameters

in	<i>time1</i>	The first time value
in	<i>time2</i>	The second time value

Return values

<i>true</i>	if the two values are equal
<i>false</i>	if the two values are not equal

Definition at line 561 of file osapi-clock.h.

References OS_time_t::ticks.

10.57.2.10 OS_TimeFromRelativeMilliseconds() `OS_time_t OS_TimeFromRelativeMilliseconds (`
 `int32 relative_msec)`

Gets an absolute time value relative to the current time.

This function adds the given interval, expressed in milliseconds, to the current clock and returns the result.

Note

This is intended to ease transitioning from a relative timeout value to an absolute timeout value. The result can be passed to any function that accepts an absolute timeout, to mimic the behavior of a relative timeout.

Parameters

in	<i>relative_msec</i>	A relative time interval, in milliseconds
----	----------------------	---

Returns

Absolute time value after adding interval

10.57.2.11 OS_TimeFromTotalMicroseconds() static `OS_time_t` OS_TimeFromTotalMicroseconds (`int64 tm`) [inline], [static]

Get an `OS_time_t` interval object from a integer number of microseconds.

This is the inverse operation of [OS_TimeGetTotalMicroseconds\(\)](#), converting the total number of microseconds into an `OS_time_t` value.

See also

[OS_TimeGetTotalMicroseconds\(\)](#)

Parameters

in	<code>tm</code>	Time interval value, in microseconds
----	-----------------	--------------------------------------

Returns

`OS_time_t` value representing the interval

Definition at line 278 of file osapi-clock.h.

References OS_TIME_TICKS_PER_USEC.

10.57.2.12 OS_TimeFromTotalMilliseconds() static `OS_time_t` OS_TimeFromTotalMilliseconds (`int64 tm`) [inline], [static]

Get an `OS_time_t` interval object from a integer number of milliseconds.

This is the inverse operation of [OS_TimeGetTotalMilliseconds\(\)](#), converting the total number of milliseconds into an `OS_time_t` value.

See also

[OS_TimeGetTotalMilliseconds\(\)](#)

Parameters

in	<code>tm</code>	Time interval value, in milliseconds
----	-----------------	--------------------------------------

Returns

`OS_time_t` value representing the interval

Definition at line 244 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC.

10.57.2.13 OS_TimeFromTotalNanoseconds() static `OS_time_t` OS_TimeFromTotalNanoseconds (`int64 tm`) [inline], [static]

Get an `OS_time_t` interval object from a integer number of nanoseconds.

This is the inverse operation of [OS_TimeGetTotalNanoseconds\(\)](#), converting the total number of nanoseconds into an `OS_time_t` value.

See also

[OS_TimeGetTotalNanoseconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value, in nanoseconds
----	-----------	-------------------------------------

Returns

[OS_time_t](#) value representing the interval

Definition at line 317 of file osapi-clock.h.

References OS_TIME_TICK_RESOLUTION_NS.

10.57.2.14 OS_TimeFromTotalSeconds() static [OS_time_t](#) OS_TimeFromTotalSeconds (

`int64 tm`) [inline], [static]

Get an [OS_time_t](#) interval object from an integer number of seconds.

This is the inverse operation of [OS_TimeGetTotalSeconds\(\)](#), converting the total number of seconds into an [OS_time_t](#) value.

See also

[OS_TimeGetTotalSeconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value, in seconds
----	-----------	---------------------------------

Returns

[OS_time_t](#) value representing the interval

Definition at line 210 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND.

10.57.2.15 OS_TimeGetFractionalPart() static `int64` OS_TimeGetFractionalPart (

[OS_time_t](#) *tm*) [inline], [static]

Get subseconds portion (fractional part only) from an [OS_time_t](#) object.

Extracts the fractional part from a given [OS_time_t](#) object. Units returned are in ticks, not normalized to any standard time unit.

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

Returns

Fractional/subsecond portion of time interval in ticks

Definition at line 333 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and [OS_time_t::ticks](#).

Referenced by OS_TimeGetMicrosecondsPart(), OS_TimeGetMillisecondsPart(), OS_TimeGetNanosecondsPart(), and OS_TimeGetSubsecondsPart().

10.57.2.16 OS_TimeGetMicrosecondsPart() static uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm) [inline], [static]

Get microseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the fractional part from a given `OS_time_t` object normalized to units of microseconds.

This function may be used to adapt applications initially implemented using an older OSAL version where `OS_time_t` was a structure containing a "seconds" and "microsecs" field.

This function will obtain a value that is compatible with the "microsecs" field of `OS_time_t` as it was defined in previous versions of OSAL, as well as the "tv_usec" field of POSIX-style "struct timeval" values.

See also

[OS_TimeGetTotalSeconds\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Number of microseconds in time interval

Definition at line 401 of file osapi-clock.h.

References OS_TIME_TICKS_PER_USEC, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.57.2.17 OS_TimeGetMillisecondsPart() static uint32 OS_TimeGetMillisecondsPart (OS_time_t tm) [inline], [static]

Get milliseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the fractional part from a given `OS_time_t` object normalized to units of milliseconds.

See also

[OS_TimeGetTotalSeconds\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Number of milliseconds in time interval

Definition at line 376 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.57.2.18 OS_TimeGetNanosecondsPart() static uint32 OS_TimeGetNanosecondsPart (OS_time_t tm) [inline], [static]

Get nanoseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the only number of nanoseconds from a given `OS_time_t` object.

This function will obtain a value that is compatible with the "tv_nsec" field of POSIX-style "struct timespec" values.

See also

[OS_TimeGetTotalSeconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

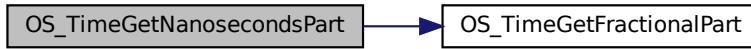
Returns

Number of nanoseconds in time interval

Definition at line 420 of file osapi-clock.h.

References OS_TIME_TICK_RESOLUTION_NS, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.57.2.19 OS_TimeGetSign() static int8_t OS_TimeGetSign (OS_time_t time) [inline], [static]

Checks the sign of the time value.

Parameters

in	<i>time</i>	The time to check
----	-------------	-------------------

Return values

-1	if the time value is negative / below 0
0	if the time value is 0
1	if the time value is positive / above 0

Definition at line 576 of file osapi-clock.h.

References OS_time_t::ticks.

Referenced by OS_TimeCompare().

10.57.2.20 OS_TimeGetSubsecondsPart() static `uint32` OS_TimeGetSubsecondsPart (
 `OS_time_t tm`) [inline], [static]

Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.

Extracts the fractional part from a given `OS_time_t` object in maximum precision, with units of 2^{-32} sec. This is a base-2 fixed-point fractional value with the point left-justified in the 32-bit value (i.e. left of MSB).

This is (mostly) compatible with the CFE "subseconds" value, where 0x80000000 represents exactly one half second, and 0 represents a full second.

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

Returns

Fractional/subsecond portion of time interval as 32-bit fixed point value

Definition at line 352 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.57.2.21 OS_TimeGetTotalMicroseconds() static `int64` OS_TimeGetTotalMicroseconds (
 `OS_time_t tm`) [inline], [static]

Get interval from an `OS_time_t` object normalized to microsecond units.

Note this refers to the complete interval, not just the fractional part.

See also

[OS_TimeFromTotalMicroseconds\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of microseconds in time interval

Definition at line 261 of file osapi-clock.h.

References OS_TIME_TICKS_PER_USEC, and OS_time_t::ticks.

10.57.2.22 OS_TimeGetTotalMilliseconds() static int64 OS_TimeGetTotalMilliseconds (
 OS_time_t tm) [inline], [static]

Get interval from an [OS_time_t](#) object normalized to millisecond units.

Note this refers to the complete interval, not just the fractional part.

See also

[OS_TimeFromTotalMilliseconds\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of milliseconds in time interval

Definition at line 227 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC, and OS_time_t::ticks.

10.57.2.23 OS_TimeGetTotalNanoseconds() static int64 OS_TimeGetTotalNanoseconds (
 OS_time_t tm) [inline], [static]

Get interval from an [OS_time_t](#) object normalized to nanosecond units.

Note this refers to the complete interval, not just the fractional part.

Note

There is no protection against overflow of the 64-bit return value. Applications must use caution to ensure that the interval does not exceed the representable range of a signed 64 bit integer - approximately 140 years.

See also

[OS_TimeFromTotalNanoseconds](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of microseconds in time interval

Definition at line 300 of file osapi-clock.h.

References OS_TIME_TICK_RESOLUTION_NS, and OS_time_t::ticks.

10.57.2.24 OS_TimeGetTotalSeconds() static int64 OS_TimeGetTotalSeconds (OS_time_t tm) [inline], [static]

Get interval from an OS_time_t object normalized to whole number of seconds.

Extracts the number of whole seconds from a given OS_time_t object, discarding any fractional component.

This may also replace a direct read of the "seconds" field from the OS_time_t object from previous versions of OSAL, where the structure was defined with separate seconds/microseconds fields.

See also

[OS_TimeGetMicrosecondsPart\(\)](#)

[OS_TimeFromTotalSeconds\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of seconds in time interval

Definition at line 193 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and OS_time_t::ticks.

10.57.2.25 OS_TimeSubtract() static OS_time_t OS_TimeSubtract (OS_time_t time1, OS_time_t time2) [inline], [static]

Computes the difference between two time intervals.

Parameters

in	time1	The first interval
in	time2	The second interval

Returns

The difference of the two intervals (time1 - time2)

Definition at line 545 of file osapi-clock.h.

References OS_time_t::ticks.

Referenced by OS_TimeCompare().

10.57.2.26 OS_TimeToRelativeMilliseconds() int32 OS_TimeToRelativeMilliseconds (OS_time_t time)

Gets a relative time value from an absolute time.

This function computes the number of milliseconds until the given absolute time value is reached in the system clock.

Note

This is intended to ease transitioning from a relative timeout value to an absolute timeout value. The result can be passed to any function that accepts a relative timeout, to mimic the behavior of an absolute timeout.

The return value of this function is intended to be compatible with the relative timeout parameter of various OSAL APIs e.g. [OS_TimedRead\(\)](#) / [OS_TimedWrite\(\)](#)

Parameters

in	<i>time</i>	An absolute time value
----	-------------	------------------------

Returns

Milliseconds until time value will be reached

Return values

<i>OS_CHECK</i>	(0) if time is the current time or is in the past
<i>OS_PEND</i>	(-1) if time is far in the future (not expressable as an int32)

10.58 OSAL Core Operation APIs

These are for OSAL core operations for startup/initialization, running, and shutdown. Typically only used in bsp's, unit tests, psp's, etc.

Functions

- void [OS_Application_Startup](#) (void)
Application startup.
- void [OS_Application_Run](#) (void)
Application run.
- int32 [OS_API_Init](#) (void)
Initialization of API.
- void [OS_API_Teardown](#) (void)
Teardown/de-initialization of OSAL 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.
- void [OS_ApplicationExit](#) (int32 Status)
Exit/Abort the application.
- int32 [OS_RegisterEventHandler](#) (OS_EventHandler_t handler)
Callback routine registration.
- size_t [OS_strlen](#) (const char *s, size_t maxlen)
get string length

10.58.1 Detailed Description

These are for OSAL core operations for startup/initialization, running, and shutdown. Typically only used in bsp's, unit tests, psp's, etc.

Not intended for user application use

10.58.2 Function Documentation

10.58.2.1 [OS_API_Init\(\)](#) int32 [OS_API_Init](#) (void)

Initialization of API.

This function returns initializes the internal data structures of the OS Abstraction Layer. It must be called in the application startup code before calling any other OS routines.

Returns

Execution status, see [OSAL Return Code Defines](#). Any error code (negative) means the OSAL can not be initialized. Typical platform specific response is to abort since additional OSAL calls will have undefined behavior.

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	Failed execution. (return value only verified in coverage test)

```
10.58.2.2 OS_API_Teardown() void OS_API_Teardown (
    void )
```

Teardown/de-initialization of OSAL API.

This is the inverse of [OS_API_Init\(\)](#). It will release all OS resources and return the system to a state similar to what it was prior to invoking [OS_API_Init\(\)](#) initially.

Normally for embedded applications, the OSAL is initialized after boot and will remain initialized in memory until the processor is rebooted. However for testing and development purposes, it is potentially useful to reset back to initial conditions.

For testing purposes, this API is designed/intended to be compatible with the [UtTest_AddTeardown\(\)](#) routine provided by the UT-Assert subsystem.

Note

This is a "best-effort" routine and it may not always be possible/guaranteed to recover all resources, particularly in the case of off-nominal conditions, or if a resource is used outside of OSAL.

For example, while this will attempt to unload all dynamically-loaded modules, doing so may not be possible and/or may induce undefined behavior if resources are in use by tasks/functions outside of OSAL.

```
10.58.2.3 OS_Application_Run() void OS_Application_Run (
    void )
```

Application run.

Run abstraction such that the same BSP can be used for operations and testing.

```
10.58.2.4 OS_Application_Startup() void OS_Application_Startup (
    void )
```

Application startup.

Startup abstraction such that the same BSP can be used for operations and testing.

```
10.58.2.5 OS_ApplicationExit() void OS_ApplicationExit (
    int32 Status )
```

Exit/Abort the application.

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.

```
10.58.2.6 OS_ApplicationShutdown() void OS_ApplicationShutdown (
    uint8 flag )
```

Initiate orderly shutdown.

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. [OS_ApplicationExit\(\)](#) which exits immediately and does not provide for any means to clean up first.

Parameters

in	flag	set to true to initiate shutdown, false to cancel
----	------	---

```
10.58.2.7 OS_DeleteAllObjects() void OS_DeleteAllObjects (
    void )
```

delete all resources created in OSAL.

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.

```
10.58.2.8 OS_IdleLoop() void OS_IdleLoop (
    void )
```

Background thread implementation - waits forever for events to occur.

This should be called from the BSP main routine or initial thread after all other board and application initialization has taken place and all other tasks are running.

Typically just waits forever until "OS_shutdown" flag becomes true.

```
10.58.2.9 OS_RegisterEventHandler() int32 OS_RegisterEventHandler (
    OS_EventHandler_t handler )
```

Callback routine registration.

This hook enables the application code to perform extra platform-specific operations on various system events such as resource creation/deletion.

Note

Some events are invoked while the resource is "locked" and therefore application-defined handlers for these events should not block or attempt to access other OSAL resources.

Parameters

in	<i>handler</i>	The application-provided event handler (must not be null)
----	----------------	---

Returns

Execution status, see [OSAL Return Code Defines](#).

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if handler is NULL

```
10.58.2.10 OS_strnlen() size_t OS_strnlen (
    const char * s,
    size_t maxlen )
```

get string length

Provides an OSAL routine to get the functionality of the (non-C99) "strnlen()" function, via the C89/C99 standard "memchr()" function instead.

Parameters

in	<i>s</i>	The input string
in	<i>maxlen</i>	Maximum length to check

Return values

<i>Length</i>	of the string or maxlen, whichever is smaller.
---------------	--

Referenced by CF_CFDP_SendMd(), and CF_WriteHistoryEntryToFile().

10.59 OSAL Condition Variable APIs

Functions

- `int32 OS_CondVarCreate (osal_id_t *var_id, const char *var_name, uint32 options)`
Creates a condition variable resource.
- `int32 OS_CondVarLock (osal_id_t var_id)`
Locks/Acquires the underlying mutex associated with a condition variable.
- `int32 OS_CondVarUnlock (osal_id_t var_id)`
Unlocks/Releases the underlying mutex associated with a condition variable.
- `int32 OS_CondVarSignal (osal_id_t var_id)`
Signals the condition variable resource referenced by var_id.
- `int32 OS_CondVarBroadcast (osal_id_t var_id)`
Broadcasts the condition variable resource referenced by var_id.
- `int32 OS_CondVarWait (osal_id_t var_id)`
Waits on the condition variable object referenced by var_id.
- `int32 OS_CondVarTimedWait (osal_id_t var_id, const OS_time_t *abs_wakeup_time)`
Time-limited wait on the condition variable object referenced by var_id.
- `int32 OS_CondVarDelete (osal_id_t var_id)`
Deletes the specified condition variable.
- `int32 OS_CondVarGetIdByName (osal_id_t *var_id, const char *var_name)`
Find an existing condition variable ID by name.
- `int32 OS_CondVarGetInfo (osal_id_t var_id, OS_condvar_prop_t *condvar_prop)`
Fill a property object buffer with details regarding the resource.

10.59.1 Detailed Description

10.59.2 Function Documentation

10.59.2.1 OS_CondVarBroadcast() `int32 OS_CondVarBroadcast (` `osal_id_t var_id)`

Broadcasts the condition variable resource referenced by var_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var_id when this function is called, all threads will be unblocked.

Note that although all threads are unblocked, because the mutex is re-acquired before the wait function returns, only a single task will be testing the condition at a given time. The order with which each blocked task runs is determined by the scheduling policy.

Parameters

in	<code>var←_id</code>	The object ID to operate on
----	----------------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condition variable

10.59.2.2 `OS_CondVarCreate()`

```
int32 OS_CondVarCreate (
    osal_id_t * var_id,
    const char * var_name,
    uint32 options )
```

Creates a condition variable resource.

A condition variable adds a more sophisticated synchronization option for mutexes, such that it can operate on arbitrary user-defined conditions rather than simply a counter or boolean (as in the case of simple semaphores).

Creating a condition variable resource in OSAL will in turn create both a basic mutex as well as a synchronization overlay. The underlying mutex is similar to the mutex functionality provided by the OSAL mutex subsystem, and can be locked and unlocked normally.

This mutex is intended to protect access to any arbitrary user-defined data object that serves as the condition being tested.

A task that needs a particular state of the object should follow this general flow:

- Lock the underlying mutex
- Test for the condition being waited for (a user-defined check on user-defined data)
- If condition IS NOT met, then call `OS_CondVarWait()` to wait, then repeat test
- If condition IS met, then unlock the underlying mutex and continue

A task that changes the state of the object should follow this general flow:

- Lock the underlying mutex
- Change the state as necessary
- Call either `OS_CondVarSignal()` or `OS_CondVarBroadcast()`
- Unlock the underlying mutex

Parameters

<code>out</code>	<code>var_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>var_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>options</code>	reserved for future use. Should be passed as 0.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if var_id or var_name are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>

Return values

<code>OS_ERR_NO_FREE_IDS</code>	if there are no more free condition variable IDs
<code>OS_ERR_NAME_TAKEN</code>	if there is already a condition variable with the same name

10.59.2.3 OS_CondVarDelete() `int32 OS_CondVarDelete (osal_id_t var_id)`

Deletes the specified condition variable.

Delete the condition variable and releases any related system resources.

Parameters

in	<code>var_id</code>	The object ID to delete
----	---------------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condvar

10.59.2.4 OS_CondVarGetIdByName() `int32 OS_CondVarGetIdByName (osal_id_t * var_id, const char * var_name)`

Find an existing condition variable ID by name.

This function tries to find an existing condition variable ID given the name. The id is returned through var_id.

Parameters

out	<code>var_id</code>	will be set to the ID of the existing resource
in	<code>var_name</code>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is var_id or var_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

```
10.59.2.5 OS_CondVarGetInfo() int32 OS_CondVarGetInfo (
    osal_id_t var_id,
    OS_condvar_prop_t * condvar_prop )
```

Fill a property object buffer with details regarding the resource.

This function will fill a structure to contain the information (name and creator) about the specified condition variable.

Parameters

in	<i>var_id</i>	The object ID to operate on
out	<i>condvar_prop</i>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the mut_prop pointer is null

```
10.59.2.6 OS_CondVarLock() int32 OS_CondVarLock (
    osal_id_t var_id )
```

Locks/Acquires the underlying mutex associated with a condition variable.

The mutex should always be locked by a task before reading or modifying the data object associated with a condition variable.

Note

This lock must be acquired by a task before invoking [OS_CondVarWait\(\)](#) or [OS_CondVarTimedWait\(\)](#) on the same condition variable.

Parameters

in	<i>var_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

10.59.2.7 OS_CondVarSignal() `int32 OS_CondVarSignal (`
 `osal_id_t var_id)`

Signals the condition variable resource referenced by var_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var_id when this function is called, one of those threads will be unblocked, as determined by the scheduling policy.

Parameters

in	<i>var_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condition variable

10.59.2.8 OS_CondVarTimedWait() `int32 OS_CondVarTimedWait (`
 `osal_id_t var_id,`
 `const OS_time_t * abs_wakeup_time)`

Time-limited wait on the condition variable object referenced by var_id.

Identical in operation to [OS_CondVarWait\(\)](#), except that the maximum amount of time that the task will be blocked is limited.

The abs_wakeup_time refers to the absolute time of the system clock at which the task should be unblocked to run, regardless of the state of the condition variable. This refers to the same system clock that is the subject of the [OS_GetLocalTime\(\)](#) API.

Parameters

in	<i>var_id</i>	The object ID to operate on
in	<i>abs_wakeup_time</i>	The system time at which the task should be unblocked (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	the id passed in is not a valid condvar

10.59.2.9 OS_CondVarUnlock() `int32 OS_CondVarUnlock (`
 `osal_id_t var_id)`

Unlocks/Releases the underlying mutex associated with a condition variable.
The mutex should be unlocked by a task once reading or modifying the data object associated with a condition variable is complete.

Parameters

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

10.59.2.10 OS_CondVarWait() `int32 OS_CondVarWait (osal_id_t var_id)`

Waits on the condition variable object referenced by var_id.

The calling task will be blocked until another task calls the function [OS_CondVarSignal\(\)](#) or [OS_CondVarBroadcast\(\)](#) on the same condition variable.

The underlying mutex associated with the condition variable must be locked and owned by the calling task at the time this function is invoked. As part of this call, the mutex will be unlocked as the task blocks. This is done in such a way that there is no possibility that another task could acquire the mutex before the calling task has actually blocked.

This atomicity with respect to blocking the task and unlocking the mutex is a critical difference between condition variables and other synchronization primitives. It avoids a window of opportunity where inherent in the simpler synchronization resource types where the state of the data could change between the time that the calling task tested the state and the time that the task actually blocks on the sync resource.

Parameters

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the id passed in is not a valid condvar

10.60 OSAL Counting Semaphore APIs

Functions

- `int32 OS_CountSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)`
Creates a counting semaphore.
- `int32 OS_CountSemGive (osal_id_t sem_id)`
Increment the semaphore value.
- `int32 OS_CountSemTake (osal_id_t sem_id)`
Decrement the semaphore value.
- `int32 OS_CountSemTimedWait (osal_id_t sem_id, uint32 msecs)`
Decrement the semaphore value with timeout.
- `int32 OS_CountSemDelete (osal_id_t sem_id)`
Deletes the specified counting Semaphore.
- `int32 OS_CountSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`
Find an existing semaphore ID by name.
- `int32 OS_CountSemGetInfo (osal_id_t sem_id, OS_count_sem_prop_t *count_prop)`
Fill a property object buffer with details regarding the resource.

10.60.1 Detailed Description

10.60.2 Function Documentation

10.60.2.1 OS_CountSemCreate() `int32 OS_CountSemCreate (`
 `osal_id_t * sem_id,`
 `const char * sem_name,`
 `uint32 sem_initial_value,`
 `uint32 options)`

Creates a counting semaphore.

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.

Note

Underlying RTOS implementations may or may not impose a specific upper limit to the value of a counting semaphore. If the OS has a specific limit and the `sem_initial_value` exceeds this limit, then `OS_INVALID_SEM_VALUE` is returned. On other implementations, any 32-bit integer value may be acceptable. For maximum portability, it is recommended to keep counting semaphore values within the range of a "short int" (i.e. between 0 and 32767). Many platforms do accept larger values, but may not be guaranteed.

Parameters

out	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>sem_name</code>	the name of the new resource to create (must not be null)
in	<code>sem_initial_value</code>	the initial value of the counting semaphore
in	<code>options</code>	Reserved for future use, should be passed as 0.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if sem name or sem_id are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if all of the semaphore ids are taken
<i>OS_ERR_NAME_TAKEN</i>	if this is already the name of a counting semaphore
<i>OS_INVALID_SEM_VALUE</i>	if the semaphore value is too high (return value only verified in coverage test)
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

10.60.2.2 OS_CountSemDelete() `int32 OS_CountSemDelete (`

`osal_id_t sem_id)`

Deletes the specified counting Semaphore.

Parameters

<i>in</i>	<i>sem_id</i>	The object ID to delete
-----------	---------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

10.60.2.3 OS_CountSemGetIdByName() `int32 OS_CountSemGetIdByName (`

`osal_id_t * sem_id,`
`const char * sem_name)`

Find an existing semaphore ID by 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

<i>out</i>	<i>sem_id</i>	will be set to the ID of the existing resource
<i>in</i>	<i>sem_name</i>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if semid or sem_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

Referenced by CF_CFDP_InitEngine().

10.60.2.4 OS_CountSemGetInfo() `int32 OS_CountSemGetInfo (`
`osal_id_t sem_id,`
`OS_count_sem_prop_t * count_prop)`

Fill a property object buffer with details regarding the resource.

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

<i>in</i>	<i>sem_id</i>	The object ID to operate on
<i>out</i>	<i>count_prop</i>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the count_prop pointer is null
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.

10.60.2.5 OS_CountSemGive() `int32 OS_CountSemGive (`
`osal_id_t sem_id)`

Increment the semaphore value.

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

<i>in</i>	<i>sem_id</i>	The object ID to operate on
-----------	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

10.60.2.6 OS_CountSemTake() `int32 OS_CountSemTake (osal_id_t sem_id)`

Decrement the semaphore value.

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	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the Id passed in is not a valid counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

10.60.2.7 OS_CountSemTimedWait() `int32 OS_CountSemTimedWait (osal_id_t sem_id, uint32 msecs)`

Decrement the semaphore value with timeout.

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	<i>sem_id</i>	The object ID to operate on
in	<i>msecs</i>	The maximum amount of time to block, in milliseconds

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_SEM_TIMEOUT</i>	if semaphore was not relinquished in time
<i>OS_ERR_INVALID_ID</i>	if the ID passed in is not a valid semaphore ID
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

Referenced by CF_CFDP_MsgOutGet().

10.61 OSAL Directory APIs

Functions

- `int32 OS_DirectoryOpen (osal_id_t *dir_id, const char *path)`
Opens a directory.
- `int32 OS_DirectoryClose (osal_id_t dir_id)`
Closes an open directory.
- `int32 OS_DirectoryRewind (osal_id_t dir_id)`
Rewinds an open directory.
- `int32 OS_DirectoryRead (osal_id_t 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.

10.61.1 Detailed Description

10.61.2 Function Documentation

10.61.2.1 OS_DirectoryClose() `int32 OS_DirectoryClose (` `osal_id_t dir_id)`

Closes an open directory.

The directory referred to by `dir_id` will be closed

Parameters

in	<code>dir_id</code>	The handle ID of the directory
----	---------------------	--------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the directory handle is invalid

Referenced by CF_CFDP_DisableEngine(), and CF_CFDP_ProcessPlaybackDirectory().

10.61.2.2 OS_DirectoryOpen() `int32 OS_DirectoryOpen (` `osal_id_t * dir_id,` `const char * path)`

Opens a directory.

Prepares for reading the files within a directory

Parameters

out	<i>dir_id</i>	Location to store handle ID of the directory (must not be null)
in	<i>path</i>	The directory to open (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dir_id or path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path argument exceeds the maximum length
<i>OS_FS_ERR_PATH_INVALID</i>	if the path argument is not valid
<i>OS_ERROR</i>	if the directory could not be opened

Referenced by CF_CFDP_PlaybackDir_Initiate().

10.61.2.3 OS_DirectoryRead() `int32 OS_DirectoryRead (`
`osal_id_t dir_id,`
`os_dirent_t * dirent)`

Reads the next name in the directory.

Obtains directory entry data for the next file from an open directory

Parameters

in	<i>dir_id</i>	The handle ID of the directory
out	<i>dirent</i>	Buffer to store directory entry information (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dirent argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the directory handle is invalid
<i>OS_ERROR</i>	at the end of the directory or if the OS call otherwise fails

Referenced by CF_CFDP_ProcessPlaybackDirectory().

10.61.2.4 OS_DirectoryRewind() `int32 OS_DirectoryRewind (`
`osal_id_t dir_id)`

Rewinds an open directory.
Resets a directory read handle back to the first file.

Parameters

in	<i>dir_id</i>	The handle ID of the directory
----	---------------	--------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the directory handle is invalid

10.61.2.5 OS_mkdir() `int32 OS_mkdir (`
 `const char * path,`
 `uint32 access)`

Makes a new directory.

Makes a directory specified by path.

Parameters

in	<i>path</i>	The new directory name (must not be null)
in	<i>access</i>	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

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path is too long to be stored locally
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_ERROR</i>	if the OS call fails (return value only verified in coverage test)

10.61.2.6 OS_rmdir() `int32 OS_rmdir (`

```
const char * path )
```

Removes a directory from the file system.

Removes a directory from the structure. The directory must be empty prior to this operation.

Parameters

in	<i>path</i>	The directory to remove
----	-------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if path is NULL
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_PATH_TOO_LONG	
OS_ERROR	if the directory remove operation failed (return value only verified in coverage test)

10.62 OSAL Return Code Defines

The specific status/return code definitions listed in this section may be extended or refined in future versions of OSAL.

Macros

- `#define OS_SUCCESS (0)`
Successful execution.
- `#define OS_ERROR (-1)`
Failed execution.
- `#define OS_INVALID_POINTER (-2)`
Invalid pointer.
- `#define OS_ERROR_ADDRESS_MISALIGNED (-3)`
Address misalignment.
- `#define OS_ERROR_TIMEOUT (-4)`
Error timeout.
- `#define OS_INVALID_INT_NUM (-5)`
Invalid interrupt number.
- `#define OS_SEM_FAILURE (-6)`
Semaphore failure.
- `#define OS_SEM_TIMEOUT (-7)`
Semaphore timeout.
- `#define OS_QUEUE_EMPTY (-8)`
Queue empty.
- `#define OS_QUEUE_FULL (-9)`
Queue full.
- `#define OS_QUEUE_TIMEOUT (-10)`
Queue timeout.
- `#define OS_QUEUE_INVALID_SIZE (-11)`
Queue invalid size.
- `#define OS_QUEUE_ID_ERROR (-12)`
Queue ID error.
- `#define OS_ERR_NAME_TOO_LONG (-13)`
name length including null terminator greater than `OS_MAX_API_NAME`
- `#define OS_ERR_NO_FREE_IDS (-14)`
No free IDs.
- `#define OS_ERR_NAME_TAKEN (-15)`
Name taken.
- `#define OS_ERR_INVALID_ID (-16)`
Invalid ID.
- `#define OS_ERR_NAME_NOT_FOUND (-17)`
Name not found.
- `#define OS_ERR_SEM_NOT_FULL (-18)`
Semaphore not full.
- `#define OS_ERR_INVALID_PRIORITY (-19)`
Invalid priority.
- `#define OS_INVALID_SEM_VALUE (-20)`
Invalid semaphore value.

- #define OS_ERR_FILE (-27)
File error.
- #define OS_ERR_NOT_IMPLEMENTED (-28)
Not implemented.
- #define OS_TIMER_ERR_INVALID_ARGS (-29)
Timer invalid arguments.
- #define OS_TIMER_ERR_TIMER_ID (-30)
Timer ID error.
- #define OS_TIMER_ERR_UNAVAILABLE (-31)
Timer unavailable.
- #define OS_TIMER_ERR_INTERNAL (-32)
Timer internal error.
- #define OS_ERR_OBJECT_IN_USE (-33)
Object in use.
- #define OS_ERR_BAD_ADDRESS (-34)
Bad address.
- #define OS_ERR_INCORRECT_OBJ_STATE (-35)
Incorrect object state.
- #define OS_ERR_INCORRECT_OBJ_TYPE (-36)
Incorrect object type.
- #define OS_ERR_STREAM_DISCONNECTED (-37)
Stream disconnected.
- #define OS_ERR_OPERATION_NOT_SUPPORTED (-38)
Requested operation not support on supplied object(s)
- #define OS_ERR_INVALID_SIZE (-40)
Invalid Size.
- #define OS_ERR_OUTPUT_TOO_LARGE (-41)
Size of output exceeds limit
- #define OS_ERR_INVALID_ARGUMENT (-42)
Invalid argument value (other than ID or size)
- #define OS_FS_ERR_PATH_TOO_LONG (-103)
FS path too long.
- #define OS_FS_ERR_NAME_TOO_LONG (-104)
FS name too long.
- #define OS_FS_ERR_DRIVE_NOT_CREATED (-106)
FS drive not created.
- #define OS_FS_ERR_DEVICE_NOT_FREE (-107)
FS device not free.
- #define OS_FS_ERR_PATH_INVALID (-108)
FS path invalid.

10.62.1 Detailed Description

The specific status/return code definitions listed in this section may be extended or refined in future versions of OSAL.

Note

Application developers should assume that any OSAL API may return any status value listed here. While the documentation of each OSAL API function indicates the return/status values that function may directly generate, functions may also pass through other status codes from related functions, so that list should not be considered absolute/exhaustive.

The `int32` data type should be used to store an OSAL status code. Negative values will always represent errors, while non-negative values indicate success. Most APIs specifically return `OS_SUCCESS` (0) upon successful execution, but some return a nonzero value, such as data size.

Ideally, in order to more easily adapt to future OSAL versions and status code extensions/refinements, applications should typically check for errors as follows:

```
int32 status;
status = OS_TaskCreate(...);  (or any other API)
if (status < OS_SUCCESS)
{
    handle or report error....
    may also check for specific codes here.
}
else
{
    handle normal/successful status...
}
```

10.62.2 Macro Definition Documentation

10.62.2.1 OS_ERR_BAD_ADDRESS #define OS_ERR_BAD_ADDRESS (-34)

Bad address.

Definition at line 124 of file osapi-error.h.

10.62.2.2 OS_ERR_FILE #define OS_ERR_FILE (-27)

File error.

Definition at line 117 of file osapi-error.h.

10.62.2.3 OS_ERR_INCORRECT_OBJ_STATE #define OS_ERR_INCORRECT_OBJ_STATE (-35)

Incorrect object state.

Definition at line 125 of file osapi-error.h.

10.62.2.4 OS_ERR_INCORRECT_OBJ_TYPE #define OS_ERR_INCORRECT_OBJ_TYPE (-36)

Incorrect object type.

Definition at line 126 of file osapi-error.h.

10.62.2.5 OS_ERR_INVALID_ARGUMENT #define OS_ERR_INVALID_ARGUMENT (-42)

Invalid argument value (other than ID or size)

Definition at line 131 of file osapi-error.h.

10.62.2.6 OS_ERR_INVALID_ID #define OS_ERR_INVALID_ID (-16)

Invalid ID.

Definition at line 112 of file osapi-error.h.

10.62.2.7 OS_ERR_INVALID_PRIORITY #define OS_ERR_INVALID_PRIORITY (-19)

Invalid priority.

Definition at line 115 of file osapi-error.h.

10.62.2.8 OS_ERR_INVALID_SIZE #define OS_ERR_INVALID_SIZE (-40)

Invalid Size.

Definition at line 129 of file osapi-error.h.

10.62.2.9 OS_ERR_NAME_NOT_FOUND #define OS_ERR_NAME_NOT_FOUND (-17)

Name not found.

Definition at line 113 of file osapi-error.h.

10.62.2.10 OS_ERR_NAME_TAKEN #define OS_ERR_NAME_TAKEN (-15)

Name taken.

Definition at line 111 of file osapi-error.h.

10.62.2.11 OS_ERR_NAME_TOO_LONG #define OS_ERR_NAME_TOO_LONG (-13)

name length including null terminator greater than [OS_MAX_API_NAME](#)

Definition at line 109 of file osapi-error.h.

10.62.2.12 OS_ERR_NO_FREE_IDS #define OS_ERR_NO_FREE_IDS (-14)

No free IDs.

Definition at line 110 of file osapi-error.h.

10.62.2.13 OS_ERR_NOT_IMPLEMENTED #define OS_ERR_NOT_IMPLEMENTED (-28)

Not implemented.

Definition at line 118 of file osapi-error.h.

10.62.2.14 OS_ERR_OBJECT_IN_USE #define OS_ERR_OBJECT_IN_USE (-33)

Object in use.

Definition at line 123 of file osapi-error.h.

10.62.2.15 OS_ERR_OPERATION_NOT_SUPPORTED #define OS_ERR_OPERATION_NOT_SUPPORTED (-38)

Requested operation not support on supplied object(s)

Definition at line 128 of file osapi-error.h.

10.62.2.16 OS_ERR_OUTPUT_TOO_LARGE #define OS_ERR_OUTPUT_TOO_LARGE (-41)
Size of output exceeds limit

Definition at line 130 of file osapi-error.h.

10.62.2.17 OS_ERR_SEM_NOT_FULL #define OS_ERR_SEM_NOT_FULL (-18)
Semaphore not full.
Definition at line 114 of file osapi-error.h.

10.62.2.18 OS_ERR_STREAM_DISCONNECTED #define OS_ERR_STREAM_DISCONNECTED (-37)
Stream disconnected.
Definition at line 127 of file osapi-error.h.

10.62.2.19 OS_ERROR #define OS_ERROR (-1)
Failed execution.
Definition at line 97 of file osapi-error.h.

10.62.2.20 OS_ERROR_ADDRESS_MISALIGNED #define OS_ERROR_ADDRESS_MISALIGNED (-3)
Address misalignment.
Definition at line 99 of file osapi-error.h.

10.62.2.21 OS_ERROR_TIMEOUT #define OS_ERROR_TIMEOUT (-4)
Error timeout.
Definition at line 100 of file osapi-error.h.

10.62.2.22 OS_FS_ERR_DEVICE_NOT_FREE #define OS_FS_ERR_DEVICE_NOT_FREE (-107)
FS device not free.
Definition at line 144 of file osapi-error.h.

10.62.2.23 OS_FS_ERR_DRIVE_NOT_CREATED #define OS_FS_ERR_DRIVE_NOT_CREATED (-106)
FS drive not created.
Definition at line 143 of file osapi-error.h.

10.62.2.24 OS_FS_ERR_NAME_TOO_LONG #define OS_FS_ERR_NAME_TOO_LONG (-104)
FS name too long.
Definition at line 142 of file osapi-error.h.

10.62.2.25 OS_FS_ERR_PATH_INVALID #define OS_FS_ERR_PATH_INVALID (-108)
FS path invalid.
Definition at line 145 of file osapi-error.h.

10.62.2.26 OS_FS_ERR_PATH_TOO_LONG #define OS_FS_ERR_PATH_TOO_LONG (-103)
FS path too long.
Definition at line 141 of file osapi-error.h.

10.62.2.27 OS_INVALID_INT_NUM #define OS_INVALID_INT_NUM (-5)
Invalid Interrupt number.
Definition at line 101 of file osapi-error.h.

10.62.2.28 OS_INVALID_POINTER #define OS_INVALID_POINTER (-2)
Invalid pointer.
Definition at line 98 of file osapi-error.h.

10.62.2.29 OS_INVALID_SEM_VALUE #define OS_INVALID_SEM_VALUE (-20)
Invalid semaphore value.
Definition at line 116 of file osapi-error.h.

10.62.2.30 OS_QUEUE_EMPTY #define OS_QUEUE_EMPTY (-8)
Queue empty.
Definition at line 104 of file osapi-error.h.

10.62.2.31 OS_QUEUE_FULL #define OS_QUEUE_FULL (-9)
Queue full.
Definition at line 105 of file osapi-error.h.

10.62.2.32 OS_QUEUE_ID_ERROR #define OS_QUEUE_ID_ERROR (-12)
Queue ID error.
Definition at line 108 of file osapi-error.h.

10.62.2.33 OS_QUEUE_INVALID_SIZE #define OS_QUEUE_INVALID_SIZE (-11)
Queue invalid size.
Definition at line 107 of file osapi-error.h.

10.62.2.34 OS_QUEUE_TIMEOUT #define OS_QUEUE_TIMEOUT (-10)
Queue timeout.
Definition at line 106 of file osapi-error.h.

10.62.2.35 OS_SEM_FAILURE #define OS_SEM_FAILURE (-6)
Semaphore failure.
Definition at line 102 of file osapi-error.h.

10.62.2.36 OS_SEM_TIMEOUT #define OS_SEM_TIMEOUT (-7)
Semaphore timeout.
Definition at line 103 of file osapi-error.h.

10.62.2.37 OS_SUCCESS #define OS_SUCCESS (0)
Successful execution.
Definition at line 96 of file osapi-error.h.

10.62.2.38 OS_TIMER_ERR_INTERNAL #define OS_TIMER_ERR_INTERNAL (-32)
Timer internal error.
Definition at line 122 of file osapi-error.h.

10.62.2.39 OS_TIMER_ERR_INVALID_ARGS #define OS_TIMER_ERR_INVALID_ARGS (-29)
Timer invalid arguments.
Definition at line 119 of file osapi-error.h.

10.62.2.40 OS_TIMER_ERR_TIMER_ID #define OS_TIMER_ERR_TIMER_ID (-30)
Timer ID error.
Definition at line 120 of file osapi-error.h.

10.62.2.41 OS_TIMER_ERR_UNAVAILABLE #define OS_TIMER_ERR_UNAVAILABLE (-31)
Timer unavailable.
Definition at line 121 of file osapi-error.h.

10.63 OSAL Error Info APIs

Functions

- static long `OS_StatusToInteger (osal_status_t Status)`
Convert a status code to a native "long" type.
- `int32 OS_GetErrorName (int32 error_num, os_err_name_t *err_name)`
Convert an error number to a string.
- `char * OS_StatusToString (osal_status_t status, os_status_string_t *status_string)`
Convert status to a string.

10.63.1 Detailed Description

10.63.2 Function Documentation

10.63.2.1 OS_GetErrorName() `int32 OS_GetErrorName (`
 `int32 error_num,`
 `os_err_name_t * err_name)`

Convert an error number to a string.

Parameters

in	<code>error_num</code>	Error number to convert
out	<code>err_name</code>	Buffer to store error string

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	if successfully converted to a string
<code>OS_INVALID_POINTER</code>	if err_name is NULL
<code>OS_ERROR</code>	if error could not be converted

10.63.2.2 OS_StatusToInteger() `static long OS_StatusToInteger (`
 `osal_status_t Status) [inline], [static]`

Convert a status code to a native "long" type.

For printing or logging purposes, this converts the given status code to a "long" (signed integer) value. It should be used in conjunction with the "%ld" conversion specifier in printf-style statements.

Parameters

in	<code>Status</code>	Execution status, see OSAL Return Code Defines
----	---------------------	--

Returns

Same status value converted to the "long" data type

Definition at line 164 of file osapi-error.h.

10.63.2.3 OS_StatusToString() `char* OS_StatusToString (`
`osal_status_t status,`
`os_status_string_t * status_string)`

Convert status to a string.

Parameters

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

Returns

Passed in string pointer

10.64 OSAL File Access Option Defines

Macros

- #define OS_READ_ONLY 0
- #define OS_WRITE_ONLY 1
- #define OS_READ_WRITE 2

10.64.1 Detailed Description

10.64.2 Macro Definition Documentation

10.64.2.1 OS_READ_ONLY #define OS_READ_ONLY 0

Read only file access

Definition at line 35 of file osapi-file.h.

10.64.2.2 OS_READ_WRITE #define OS_READ_WRITE 2

Read write file access

Definition at line 37 of file osapi-file.h.

10.64.2.3 OS_WRITE_ONLY #define OS_WRITE_ONLY 1

Write only file access

Definition at line 36 of file osapi-file.h.

10.65 OSAL Reference Point For Seek Offset Defines

Macros

- #define OS_SEEK_SET 0
- #define OS_SEEK_CUR 1
- #define OS_SEEK_END 2

10.65.1 Detailed Description

10.65.2 Macro Definition Documentation

10.65.2.1 OS_SEEK_CUR #define OS_SEEK_CUR 1

Seek offset current

Definition at line 44 of file osapi-file.h.

10.65.2.2 OS_SEEK_END #define OS_SEEK_END 2

Seek offset end

Definition at line 45 of file osapi-file.h.

10.65.2.3 OS_SEEK_SET #define OS_SEEK_SET 0

Seek offset set

Definition at line 43 of file osapi-file.h.

10.66 OSAL Standard File APIs

Functions

- `int32 OS_OpenCreate (osal_id_t *filedes, const char *path, int32 flags, int32 access_mode)`
Open or create a file.
- `int32 OS_close (osal_id_t filedes)`
Closes an open file handle.
- `int32 OS_read (osal_id_t filedes, void *buffer, size_t nbytes)`
Read from a file handle.
- `int32 OS_write (osal_id_t filedes, const void *buffer, size_t nbytes)`
Write to a file handle.
- `int32 OS_TimedReadAbs (osal_id_t filedes, void *buffer, size_t nbytes, OS_time_t abstime)`
File/Stream input read with a timeout.
- `int32 OS_TimedRead (osal_id_t filedes, void *buffer, size_t nbytes, int32 timeout)`
File/Stream input read with a timeout.
- `int32 OS_TimedWriteAbs (osal_id_t filedes, const void *buffer, size_t nbytes, OS_time_t abstime)`
File/Stream output write with a timeout.
- `int32 OS_TimedWrite (osal_id_t filedes, const void *buffer, size_t nbytes, int32 timeout)`
File/Stream output write with a timeout.
- `int32 OS_chmod (const char *path, uint32 access_mode)`
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 (osal_id_t 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 (osal_id_t 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.

10.66.1 Detailed Description

10.66.2 Function Documentation

```
10.66.2.1 OS_chmod() int32 OS_chmod (
    const char * path,
    uint32 access_mode )
```

Changes the permissions of a file.

Parameters

in	<i>path</i>	File to change (must not be null)
in	<i>access_mode</i>	Desired access mode - see OSAL File Access Option Defines

Note

Some file systems do not implement permissions. If the underlying OS does not support this operation, then [OS_ERR_NOT_IMPLEMENTED](#) is returned.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution. (return value only verified in coverage test)
OS_ERR_NOT_IMPLEMENTED	if the filesystem does not support this call
OS_INVALID_POINTER	if the path argument is NULL

```
10.66.2.2 OS_close() int32 OS_close (
    osal_id_t filedes )
```

Closes an open file handle.

This closes regular file handles and any other file-like resource, such as network streams or pipes.

Parameters

in	<i>filedes</i>	The handle ID to operate on
----	----------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
OS_ERROR	if an unexpected/unhandled error occurs (return value only verified in coverage test)

Referenced by CF_WrappedClose().

```
10.66.2.3 OS_CloseAllFiles() int32 OS_CloseAllFiles (
```

```
    void  )
```

Close all open files.

Closes All open files that were opened through the OSAL

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	if one or more file close returned an error (return value only verified in coverage test)

10.66.2.4 OS_CloseFileByName() `int32 OS_CloseFileByName (`
 `const char * Filename)`

Close a file by 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	<i>Filename</i>	The file to close (must not be null)
----	-----------------	--------------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_FS_ERR_PATH_INVALID	if the file is not found
OS_ERROR	if the file close returned an error (return value only verified in coverage test)
OS_INVALID_POINTER	if the filename argument is NULL

10.66.2.5 OS_cp() `int32 OS_cp (`
 `const char * src,`
 `const char * dest)`

Copies a single file from src to dest.

Note

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

Parameters

in	<i>src</i>	The source file to operate on (must not be null)
in	<i>dest</i>	The destination file (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be accessed
<i>OS_INVALID_POINTER</i>	if src or dest are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the dest name is too long to be stored locally

10.66.2.6 OS_FDGetInfo() `int32 OS_FDGetInfo (osal_id_t filedes, OS_file_prop_t * fd_prop)`

Obtain information about an open file.

Copies the information of the given file descriptor into a structure passed in

Parameters

in	<i>filedes</i>	The handle ID to operate on
out	<i>fd_prop</i>	Storage buffer for file information (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_INVALID_POINTER</i>	if the fd_prop argument is NULL

10.66.2.7 OS_FileOpenCheck() `int32 OS_FileOpenCheck (const char * Filename)`

Checks to see if a file is open.

This function takes a filename and determines if the file is open. The function will return success if the file is open.

Parameters

in	<i>Filename</i>	The file to operate on (must not be null)
----	-----------------	---

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	if the file is open
<i>OS_ERROR</i>	if the file is not open
<i>OS_INVALID_POINTER</i>	if the filename argument is NULL

Referenced by CF_CFDP_S_SubstateSendMetadata().

10.66.2.8 OS_lseek() `int32 OS_lseek (`
`osal_id_t filedes,`
`int32 offset,`
`uint32 whence)`

Seeks to the specified position of an open file.

Sets the read/write pointer to a specific offset in a specific file.

Parameters

in	<i>filedes</i>	The handle ID to operate on
in	<i>offset</i>	The file offset to seek to
in	<i>whence</i>	The reference point for offset, see OSAL Reference Point For Seek Offset Defines

Returns

Byte offset from the beginning of the file or appropriate error code, see [OSAL Return Code Defines](#)

Return values

<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)

Referenced by CF_WrappedLseek().

10.66.2.9 OS_mv() `int32 OS_mv (`
`const char * src,`
`const char * dest)`

Move a single file from src to 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.

Note

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

Parameters

in	<i>src</i>	The source file to operate on (must not be null)
in	<i>dest</i>	The destination file (must not be null)

ReturnsExecution status, see [OSAL Return Code Defines](#)**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be renamed.
<i>OS_INVALID_POINTER</i>	if src or dest are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the dest name is too long to be stored locally

Referenced by CF_CFDP_MoveFile(), and CF_CFDP_R2_RecvMd().

```
10.66.2.10 OS_OpenCreate() int32 OS_OpenCreate (
    osal_id_t * filedes,
    const char * path,
    int32 flags,
    int32 access_mode )
```

Open or create a file.

Implements the same as OS_open/OS_creat but follows the OSAL paradigm of outputting the ID/descriptor separately from the return value, rather than relying on the user to convert it back.

Parameters

out	<i>filedes</i>	The handle ID (OS_OBJECT_ID_UNDEFINED on failure) (must not be null)
in	<i>path</i>	File name to create or open (must not be null)
in	<i>flags</i>	The file permissions - see OS_file_flag_t
in	<i>access_mode</i>	Intended access mode - see OSAL File Access Option Defines

ReturnsExecution status, see [OSAL Return Code Defines](#)**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the command was not executed properly
<i>OS_INVALID_POINTER</i>	if pointer argument was NULL
<i>OS_ERR_NO_FREE_IDS</i>	if all available file handles are in use
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the filename portion of the path exceeds OS_MAX_FILE_NAME
<i>OS_FS_ERR_PATH_INVALID</i>	if the path argument is not valid
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path argument exceeds OS_MAX_PATH_LEN

Referenced by CF_WrappedOpenCreate().

```
10.66.2.11 OS_read() int32 OS_read (
    osal_id_t filedes,
    void * buffer,
    size_t nbytes )
```

Read from a file handle.

Reads up to nbytes from a file, and puts them into buffer.

If the file position is at the end of file (or beyond, if the OS allows) then this function will return 0.

Parameters

in	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)

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

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

Return values

<i>OS_INVALID_POINTER</i>	if buffer is a null pointer
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
0	if at end of file/stream data

Referenced by CF_WrappedRead().

```
10.66.2.12 OS_remove() int32 OS_remove (
    const char * path )
```

Removes a file from the file system.

Removes a given filename from the drive

Note

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

Parameters

in	<i>path</i>	The file to operate on (must not be null)
----	-------------	---

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if there is no device or the driver returns error
<i>OS_INVALID_POINTER</i>	if path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if path is too long to be stored locally
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the name of the file to remove is too long

Referenced by CF_CFDP_HandleNotKeepFile(), and CF_CFDP_MoveFile().

```
10.66.2.13 OS_rename() int32 OS_rename (
    const char * old_filename,
    const char * new_filename )
```

Renames a file.

Changes the name of a file, where the source and destination reside on the same file system.

Note

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

Parameters

in	<i>old_filename</i>	The original filename (must not be null)
in	<i>new_filename</i>	The desired filename (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be opened or renamed.
<i>OS_INVALID_POINTER</i>	if old_filename or new_filename are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the new name is too long to be stored locally

```
10.66.2.14 OS_stat() int32 OS_stat (
    const char * path,
```

```
    os_fstat_t * filestats )
```

Obtain information about a file or directory.

Returns information about a file or directory in an `os_fstat_t` structure

Parameters

in	<i>path</i>	The file to operate on (must not be null)
out	<i>filestats</i>	Buffer to store file information (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if path or filestats is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the path is too long to be stored locally
<code>OS_FS_ERR_NAME_TOO_LONG</code>	if the name of the file is too long to be stored
<code>OS_FS_ERR_PATH_INVALID</code>	if path cannot be parsed
<code>OS_ERROR</code>	if the OS call failed

10.66.2.15 `OS_TimedRead()` `int32 OS_TimedRead(`

```
    osal_id_t filedes,
    void * buffer,
    size_t nbytes,
    int32 timeout )
```

File/Stream input read with a 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, such as pipes or special devices.

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 the file position is at the end of file or end of stream data (e.g. if the remote end has closed the connection), then this function will immediately return 0 without blocking for the timeout period.

If no data is immediately available, but the underlying resource/stream is still connected to a peer, this will wait up to the given timeout for additional data to appear. If no data appears within the timeout period, then this returns the `OS_ERROR_TIMEOUT` status code. This allows the caller to differentiate an open (but idle) socket connection from a connection which has been closed by the remote peer.

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	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>timeout</i>	Maximum time to wait, in milliseconds, relative to current time (OS_PEND = forever)

Returns

Byte count on success or appropriate error code, see [OSAL Return Code Defines](#)

Return values

<i>OS_ERROR_TIMEOUT</i>	if no data became available during timeout period
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is not valid
<i>0</i>	if at end of file/stream data

10.66.2.16 OS_TimedReadAbs()

```
int32 OS_TimedReadAbs (
    osal_id_t filedes,
    void * buffer,
    size_t nbytes,
    OS_time_t abstime )
```

File/Stream input read with a 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, such as pipes or special devices.

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 the file position is at the end of file or end of stream data (e.g. if the remote end has closed the connection), then this function will immediately return 0 without blocking for the timeout period.

If no data is immediately available, but the underlying resource/stream is still connected to a peer, this will wait up to the given timeout for additional data to appear. If no data appears within the timeout period, then this returns the *OS_ERROR_TIMEOUT* status code. This allows the caller to differentiate an open (but idle) socket connection from a connection which has been closed by the remote peer.

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	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>abstime</i>	Absolute time at which this function should return, if no data is readable

Returns

Byte count on success or appropriate error code, see [OSAL Return Code Defines](#)

Return values

<i>OS_ERROR_TIMEOUT</i>	if no data became available during timeout period
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is not valid
<i>0</i>	if at end of file/stream data

```
10.66.2.17 OS_TimedWrite() int32 OS_TimedWrite (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes,
    int32 timeout )
```

File/Stream output write with a 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	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>timeout</i>	Maximum time to wait, in milliseconds, relative to current time (OS_PEND = forever)

Returns

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

Return values

OS_ERROR_TIMEOUT	if no data became available during timeout period
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
OS_ERR_INVALID_SIZE	if the passed-in size is not valid
OS_INVALID_POINTER	if the passed-in buffer is not valid
0	if file/stream cannot accept any more data

```
10.66.2.18 OS_TimedWriteAbs() int32 OS_TimedWriteAbs (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes,
    OS_time_t abstime )
```

File/Stream output write with a 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	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>abstime</i>	Absolute time at which this function should return, if no data is readable

Returns

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

Return values

OS_ERROR_TIMEOUT	if no data became available during timeout period
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
OS_ERR_INVALID_SIZE	if the passed-in size is not valid
OS_INVALID_POINTER	if the passed-in buffer is not valid
0	if file/stream cannot accept any more data

```
10.66.2.19 OS_write() int32 OS_write (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes )
```

Write to a file handle.

Writes to a file. copies up to a maximum of nbytes of buffer to the file described in filedes

Parameters

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)

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

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

Return values

OS_INVALID_POINTER	if buffer is NULL
------------------------------------	-------------------

Return values

<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>0</i>	if file/stream cannot accept any more data

Referenced by CF_WrappedWrite().

10.67 OSAL File System Level APIs

Functions

- `int32 OS_FileSysAddFixedMap (osal_id_t *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, size_t blocksize, osal_blockcount_t 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, size_t blocksize, osal_blockcount_t 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_FileSysStatVolume (const char *name, OS_statvfs_t *statbuf)`
Obtains information about size and free space 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 an OSAL Virtual file system path to a host Local path.
- `int32 OS_GetFsInfo (os_fsinfo_t *filesys_info)`
Returns information about the file system.

10.67.1 Detailed Description

10.67.2 Function Documentation

10.67.2.1 OS_chkfs() `int32 OS_chkfs (`

```
    const char * name,
    bool repair )
```

Checks the health of a file system and repairs it if necessary.

Checks the drives for inconsistencies and optionally also repairs it

Note

not all operating systems implement this function. If the underlying OS does not provide a facility to check the volume, then OS_ERR_NOT_IMPLEMENTED will be returned.

Parameters

in	<code>name</code>	The device/path to operate on (must not be null)
in	<code>repair</code>	Whether to also repair inconsistencies

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution. (return value only verified in coverage test)
<i>OS_INVALID_POINTER</i>	Name is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the name is too long
<i>OS_ERROR</i>	Failed execution. (return value only verified in coverage test)

```
10.67.2.2 OS_FileSysAddFixedMap() int32 OS_FileSysAddFixedMap (
    osal_id_t * filesystem_id,
    const char * phys_path,
    const char * virt_path )
```

Create a fixed mapping between an existing directory and a virtual OSAL mount point.

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 application.

Note

OSAL virtual mount points are required to be a single, non-empty top-level directory name. Virtual path names always follow the form /<virt_mount_point>/<relative_path>/<file>. Only the relative path may be omitted/empty (i.e. /<virt_mount_point>/<file>) but the virtual mount point must be present and not an empty string. In particular this means it is not possible to directly refer to files in the "root" of the native file system from OSAL. However it is possible to create a virtual map to the root, such as by calling:

```
OS_FileSysAddFixedMap (&fs_id, "/", "/root");
```

Parameters

out	<i>filesystem_id</i>	A buffer to store the ID of the file system mapping (must not be null)
in	<i>phys_path</i>	The native system directory (an existing mount point) (must not be null)
in	<i>virt_path</i>	The virtual mount point of this filesystem (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the overall <i>phys_path</i> is too long
<i>OS_ERR_NAME_TOO_LONG</i>	if the <i>phys_path</i> basename (filesystem name) is too long
<i>OS_INVALID_POINTER</i>	if any argument is NULL

```
10.67.2.3 OS_FileSysStatVolume() int32 OS_FileSysStatVolume (
    const char * name,
    OS_statvfs_t * statbuf )
```

Obtains information about size and free space in a volume.

Populates the supplied `OS_statvfs_t` structure, which includes the block size and total/free blocks in a file system volume.

This replaces two older OSAL calls:

`OS_fsBlocksFree()` is determined by reading the `blocks_free` output struct member `OS_fsBytesFree()` is determined by multiplying `blocks_free` by the `block_size` member

Parameters

in	<i>name</i>	The device/path to operate on (must not be null)
out	<i>statbuf</i>	Output structure to populate (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if name or statbuf is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_ERROR</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.67.2.4 OS_FS_GetPhysDriveName() int32 OS_FS_GetPhysDriveName (
```

```
    char * PhysDriveName,
    const char * MountPoint )
```

Obtains the physical drive name associated with a mount point.

Returns the name of the physical volume associated with the drive, when given the OSAL mount point of the drive

Parameters

out	<i>PhysDriveName</i>	Buffer to store physical drive name (must not be null)
in	<i>MountPoint</i>	OSAL mount point (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if either parameter is NULL
<code>OS_ERR_NAME_NOT_FOUND</code>	if the MountPoint is not mounted in OSAL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the MountPoint is too long

10.67.2.5 OS_GetFsInfo() `int32 OS_GetFsInfo (`
 `os_fsinfo_t * filesys_info)`

Returns information about the file system.

Returns information about the file system in an `os_fsinfo_t`. This includes the number of open files and file systems

Parameters

<code>out</code>	<code>filesys_info</code>	Buffer to store filesystem information (must not be null)
------------------	---------------------------	---

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>filesys_info</code> is NULL

10.67.2.6 OS_initfs() `int32 OS_initfs (`
 `char * address,`
 `const char * devname,`
 `const char * volname,`
 `size_t blocksize,`
 `osal_blockcount_t numblocks)`

Initializes an existing file system.

Initializes a file system on the target.

Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RA←M0","RAM1", etc if multiple devices are created. The underlying implementation uses this to select the correct filesystem type/format, and this may also be used to differentiate between RAM disks and real physical disks.

Parameters

<code>in</code>	<code>address</code>	The address at which to start the new disk. If <code>address == 0</code> , then space will be allocated by the OS
<code>in</code>	<code>devname</code>	The underlying kernel device to use, if applicable. (must not be null)
<code>in</code>	<code>volname</code>	The name of the volume (see note) (must not be null)
<code>in</code>	<code>blocksize</code>	The size of a single block on the drive
<code>in</code>	<code>numblocks</code>	The number of blocks to allocate for the drive

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if devname or volname are NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the name is too long
<i>OS_FS_ERR_DEVICE_NOT_FREE</i>	if the volume table is full
<i>OS_FS_ERR_DRIVE_NOT_CREATED</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.67.2.7 OS_mkfs() int32 OS_mkfs (
    char * address,
    const char * devname,
    const char * volname,
    size_t blocksize,
    osal_blockcount_t numblocks )
```

Makes a file system on the target.

Makes a file system on the target. Highly dependent on underlying OS and dependent on OS volume table definition.

Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RA←M0","RAM1", etc if multiple devices are created. The underlying implementation uses this to select the correct filesystem type/format, and this may also be used to differentiate between RAM disks and real physical disks.

Parameters

in	<i>address</i>	The address at which to start the new disk. If address == 0 space will be allocated by the OS.
in	<i>devname</i>	The underlying kernel device to use, if applicable. (must not be null)
in	<i>volname</i>	The name of the volume (see note) (must not be null)
in	<i>blocksize</i>	The size of a single block on the drive
in	<i>numblocks</i>	The number of blocks to allocate for the drive

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if devname or volname is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the overall devname or volname is too long
<i>OS_FS_ERR_DEVICE_NOT_FREE</i>	if the volume table is full
<i>OS_FS_ERR_DRIVE_NOT_CREATED</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.67.2.8 OS_mount() int32 OS_mount (
```

```
    const char * devname,
    const char * mountpoint )
```

Mounts a file system.

Mounts a file system / block device at the given mount point.

Parameters

in	<i>devname</i>	The name of the drive to mount. devname is the same from OS_mkfs (must not be null)
in	<i>mountpoint</i>	The name to call this disk from now on (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_NAME_NOT_FOUND	if the device name does not exist in OSAL
OS_FS_ERR_PATH_TOO_LONG	if the mount point string is too long
OS_INVALID_POINTER	if any argument is NULL
OS_ERROR	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.67.2.9 OS_rmfs()

```
int32 OS_rmfs(
```

```
    const char * devname )
```

Removes a file system.

This function will remove or un-map the target file system. Note that this is not the same as un-mounting the file system.

Parameters

in	<i>devname</i>	The name of the "generic" drive (must not be null)
----	----------------	--

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if devname is NULL
OS_FS_ERR_PATH_TOO_LONG	if the devname is too long
OS_ERR_NAME_NOT_FOUND	if the devname does not exist in OSAL
OS_ERROR	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.67.2.10 OS_TranslatePath() int32 OS_TranslatePath (
    const char * VirtualPath,
    char * LocalPath )
```

Translates an OSAL Virtual file system path to a host Local path.
 Translates a virtual path to an actual system path name

Note

The buffer provided in the LocalPath argument is required to be at least OS_MAX_PATH_LEN characters in length.

Parameters

in	<i>VirtualPath</i>	OSAL virtual path name (must not be null)
out	<i>LocalPath</i>	Buffer to store native/translated path name (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if either parameter is NULL
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the filename component is too long
<i>OS_FS_ERR_PATH_INVALID</i>	if either parameter cannot be interpreted as a path
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if either input or output pathnames are too long

```
10.67.2.11 OS_unmount() int32 OS_unmount (
```

```
    const char * mountpoint )
```

Unmounts a mounted file system.

This function will unmount a drive from the file system and make all open file descriptors useless.

Note

Any open file descriptors referencing this file system should be closed prior to unmounting a drive

Parameters

in	<i>mountpoint</i>	The mount point to remove from OS_mount (must not be null)
----	-------------------	--

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if name is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the absolute path given is too long

Return values

<i>OS_ERR_NAME_NOT_FOUND</i>	if the mountpoint is not mounted in OSAL
<i>OS_ERROR</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.68 OSAL Heap APIs

Functions

- `int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)`

Return current info on the heap.

10.68.1 Detailed Description

10.68.2 Function Documentation

10.68.2.1 `OS_HeapGetInfo()` `int32 OS_HeapGetInfo (` `OS_heap_prop_t * heap_prop)`

Return current info on the heap.

Parameters

<code>out</code>	<code>heap_prop</code>	Storage buffer for heap info
------------------	------------------------	------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if the heap_prop argument is NULL

10.69 OSAL Object Type Defines

Macros

- `#define OS_OBJECT_TYPE_UNDEFINED 0x00`
Object type undefined.
- `#define OS_OBJECT_TYPE_OS_TASK 0x01`
Object task type.
- `#define OS_OBJECT_TYPE_OS_QUEUE 0x02`
Object queue type.
- `#define OS_OBJECT_TYPE_OS_COUNTSEM 0x03`
Object counting semaphore type.
- `#define OS_OBJECT_TYPE_OS_BINSEM 0x04`
Object binary semaphore type.
- `#define OS_OBJECT_TYPE_OS_MUTEX 0x05`
Object mutex type.
- `#define OS_OBJECT_TYPE_OS_STREAM 0x06`
Object stream type.
- `#define OS_OBJECT_TYPE_OS_DIR 0x07`
Object directory type.
- `#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08`
Object timebase type.
- `#define OS_OBJECT_TYPE_OS_TIMECB 0x09`
Object timer callback type.
- `#define OS_OBJECT_TYPE_OS_MODULE 0x0A`
Object module type.
- `#define OS_OBJECT_TYPE_OS_FILESYS 0x0B`
Object file system type.
- `#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C`
Object console type.
- `#define OS_OBJECT_TYPE_OS_CONDVAR 0x0D`
Object condition variable type.
- `#define OS_OBJECT_TYPE_USER 0x10`
Object user type.

10.69.1 Detailed Description

10.69.2 Macro Definition Documentation

10.69.2.1 OS_OBJECT_TYPE_OS_BINSEM `#define OS_OBJECT_TYPE_OS_BINSEM 0x04`
Object binary semaphore type.
Definition at line 42 of file osapi-idmap.h.

10.69.2.2 OS_OBJECT_TYPE_OS_CONDVAR `#define OS_OBJECT_TYPE_OS_CONDVAR 0x0D`
Object condition variable type.
Definition at line 51 of file osapi-idmap.h.

10.69.2.3 OS_OBJECT_TYPE_OS_CONSOLE #define OS_OBJECT_TYPE_OS_CONSOLE 0x0C
Object console type.
Definition at line 50 of file osapi-idmap.h.

10.69.2.4 OS_OBJECT_TYPE_OS_COUNTSEM #define OS_OBJECT_TYPE_OS_COUNTSEM 0x03
Object counting semaphore type.
Definition at line 41 of file osapi-idmap.h.

10.69.2.5 OS_OBJECT_TYPE_OS_DIR #define OS_OBJECT_TYPE_OS_DIR 0x07
Object directory type.
Definition at line 45 of file osapi-idmap.h.

10.69.2.6 OS_OBJECT_TYPE_OS_FILESYS #define OS_OBJECT_TYPE_OS_FILESYS 0x0B
Object file system type.
Definition at line 49 of file osapi-idmap.h.

10.69.2.7 OS_OBJECT_TYPE_OS_MODULE #define OS_OBJECT_TYPE_OS_MODULE 0x0A
Object module type.
Definition at line 48 of file osapi-idmap.h.

10.69.2.8 OS_OBJECT_TYPE_OS_MUTEX #define OS_OBJECT_TYPE_OS_MUTEX 0x05
Object mutex type.
Definition at line 43 of file osapi-idmap.h.

10.69.2.9 OS_OBJECT_TYPE_OS_QUEUE #define OS_OBJECT_TYPE_OS_QUEUE 0x02
Object queue type.
Definition at line 40 of file osapi-idmap.h.

10.69.2.10 OS_OBJECT_TYPE_OS_STREAM #define OS_OBJECT_TYPE_OS_STREAM 0x06
Object stream type.
Definition at line 44 of file osapi-idmap.h.

10.69.2.11 OS_OBJECT_TYPE_OS_TASK #define OS_OBJECT_TYPE_OS_TASK 0x01
Object task type.
Definition at line 39 of file osapi-idmap.h.

10.69.2.12 OS_OBJECT_TYPE_OS_TIMEBASE #define OS_OBJECT_TYPE_OS_TIMEBASE 0x08
Object timebase type.
Definition at line 46 of file osapi-idmap.h.

10.69.2.13 OS_OBJECT_TYPE_OS_TIMECB #define OS_OBJECT_TYPE_OS_TIMECB 0x09
Object timer callback type.
Definition at line 47 of file osapi-idmap.h.

10.69.2.14 OS_OBJECT_TYPE_UNDEFINED #define OS_OBJECT_TYPE_UNDEFINED 0x00
Object type undefined.
Definition at line 38 of file osapi-idmap.h.

10.69.2.15 OS_OBJECT_TYPE_USER #define OS_OBJECT_TYPE_USER 0x10
Object user type.
Definition at line 52 of file osapi-idmap.h.

10.70 OSAL Object ID Utility APIs

Functions

- static unsigned long [OS_ObjectIdToInteger](#) ([osal_id_t](#) object_id)
Obtain an integer value corresponding to an object ID.
- static [osal_id_t OS_ObjectIdFromInteger](#) (unsigned long value)
Obtain an osal ID corresponding to an integer value.
- static bool [OS_ObjectIdEqual](#) ([osal_id_t](#) object_id1, [osal_id_t](#) object_id2)
Check two OSAL object ID values for equality.
- static bool [OS_ObjectIdDefined](#) ([osal_id_t](#) object_id)
Check if an object ID is defined.
- int32 [OS_GetResourceName](#) ([osal_id_t](#) object_id, char *buffer, size_t buffer_size)
Obtain the name of an object given an arbitrary object ID.
- [osal_objtype_t OS_IdentifyObject](#) ([osal_id_t](#) object_id)
Obtain the type of an object given an arbitrary object ID.
- int32 [OS_ConvertToArrayIndex](#) ([osal_id_t](#) object_id, [osal_index_t](#) *ArrayIndex)
Converts an abstract ID into a number suitable for use as an array index.
- int32 [OS_ObjectIdToArrayIndex](#) ([osal_objtype_t](#) idtype, [osal_id_t](#) object_id, [osal_index_t](#) *ArrayIndex)
Converts an abstract ID into a number suitable for use as an array index.
- void [OS_ForEachObject](#) ([osal_id_t](#) creator_id, [OS_ArgCallback_t](#) callback_ptr, void *callback_arg)
call the supplied callback function for all valid object IDs
- void [OS_ForEachObjectType](#) ([osal_objtype_t](#) objtype, [osal_id_t](#) creator_id, [OS_ArgCallback_t](#) callback_ptr, void *callback_arg)
call the supplied callback function for valid object IDs of a specific type

10.70.1 Detailed Description

10.70.2 Function Documentation

10.70.2.1 OS_ConvertToArrayIndex() [int32 OS_ConvertToArrayIndex](#) (
 [osal_id_t](#) object_id,
 [osal_index_t](#) * ArrayIndex)

Converts an abstract ID into a number suitable for use as an array index.

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.

This routine accepts any object type, and returns a value based on the maximum number of objects for that type. This is equivalent to invoking [OS_ObjectIdToArrayIndex\(\)](#) with the idtype set to [OS_OBJECT_TYPE_UNDEFINED](#).

See also

[OS_ObjectIdToArrayIndex](#)

Parameters

in	object_id	The object ID to operate on
out	*ArrayIndex	The Index to return (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the object_id argument is not valid
<code>OS_INVALID_POINTER</code>	if the ArrayIndex is NULL

10.70.2.2 OS_ForEachObject() `void OS_ForEachObject (`
 `osal_id_t creator_id,`
 `OS_ArgCallback_t callback_ptr,`
 `void * callback_arg)`

call the supplied callback function for all valid object IDs

Loops through all defined OSAL objects of all types and calls callback_ptr on each one If creator_id is nonzero then only objects with matching creator id are processed.

Parameters

in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

10.70.2.3 OS_ForEachObjectType() `void OS_ForEachObjectType (`
 `osal_objtype_t objtype,`
 `osal_id_t creator_id,`
 `OS_ArgCallback_t callback_ptr,`
 `void * callback_arg)`

call the supplied callback function for valid object IDs of a specific type

Loops through all defined OSAL objects of a specific type and calls callback_ptr on each one If creator_id is nonzero then only objects with matching creator id are processed.

Parameters

in	<i>objtype</i>	The type of objects to iterate
in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

10.70.2.4 OS_GetResourceName() `int32 OS_GetResourceName (`
 `osal_id_t object_id,`
 `char * buffer,`

```
size_t buffer_size )
```

Obtain the name of an object given an arbitrary object ID.

All OSAL resources generally have a name associated with them. This allows application code to retrieve the name of any valid OSAL object ID.

Parameters

in	<i>object_id</i>	The object ID to operate on
out	<i>buffer</i>	Buffer in which to store the name (must not be null)
in	<i>buffer_size</i>	Size of the output storage buffer (must not be zero)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the passed-in ID is not a valid OSAL ID
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is invalid
<i>OS_ERR_NAME_TOO_LONG</i>	if the name will not fit in the buffer provided

10.70.2.5 OS_IdentifyObject()

```
osal_objtype_t OS_IdentifyObject (
    osal_id_t object_id )
```

Obtain the type of an object given an arbitrary object ID.

Given an arbitrary object ID, get the type of the object

Parameters

in	<i>object_id</i>	The object ID to operate on
----	------------------	-----------------------------

Returns

The object type portion of the object_id, see [OSAL Object Type Defines](#) for expected values

10.70.2.6 OS_ObjectIdDefined()

```
static bool OS_ObjectIdDefined (
    osal_id_t object_id ) [inline], [static]
```

Check if an object ID is defined.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This returns false if the ID is NOT a defined resource (i.e. free/empty/invalid).

Note

OS_ObjectIdDefined(OS_OBJECT_ID_UNDEFINED) is always guaranteed to be false.

Parameters

in	<i>object_id</i>	The first object ID
----	------------------	---------------------

Definition at line 150 of file osapi-idmap.h.

References OS_ObjectIdToInteger().

Referenced by CF_CFDP_CloseFiles(), CF_CFDP_MsgOutGet(), CF_CFDP_ResetTransaction(), and CF_CFDP_SubstateSendMetadata().

10.70.2.7 OS_ObjectIdEqual() static bool OS_ObjectIdEqual (

```
    osal_id_t object_id1,  
    osal_id_t object_id2 ) [inline], [static]
```

Check two OSAL object ID values for equality.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This checks two values for equality, replacing the "==" operator.

Parameters

in	<i>object_id1</i>	The first object ID
in	<i>object_id2</i>	The second object ID

Returns

true if the object IDs are equal

Definition at line 129 of file osapi-idmap.h.

References OS_ObjectIdToInteger().

10.70.2.8 OS_ObjectIdFromInteger() static osal_id_t OS_ObjectIdFromInteger (

```
    unsigned long value ) [inline], [static]
```

Obtain an osal ID corresponding to an integer value.

Provides the inverse of [OS_ObjectIdToInteger\(\)](#). Reconstitutes the original osal_id_t type from an integer representation.

Parameters

in	<i>value</i>	The integer representation of an OSAL ID
----	--------------	--

Returns

The ID value converted to an osal_id_t

Definition at line 102 of file osapi-idmap.h.

10.70.2.9 OS_ObjectIdToArrayIndex() int32 OS_ObjectIdToArrayIndex (

```
    osal_objtype_t idtype,  
    osal_id_t object_id,  
    osal_index_t * ArrayIndex )
```

Converts an abstract ID into a number suitable for use as an array index.

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.

This routine operates on a specific object type, and returns a value based on the maximum number of objects for that type.

If the idtype is passed as [OS_OBJECT_TYPE_UNDEFINED](#), then object type verification is skipped and any object ID will be accepted and converted to an index. In this mode, the range of the output depends on the actual passed-in object type.

If the idtype is passed as any other value, the passed-in ID value is first confirmed to be the correct type. This check will guarantee that the output is within an expected range; for instance, if the type is passed as [OS_OBJECT_TYPE_OS_TASK](#), then the output index is guaranteed to be between 0 and [OS_MAX_TASKS](#)-1 after successful conversion.

Parameters

in	<i>idtype</i>	The object type to convert
in	<i>object_id</i>	The object ID to operate on
out	<i>*ArrayIndex</i>	The Index to return (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the object_id argument is not valid
OS_INVALID_POINTER	if the ArrayIndex is NULL

10.70.2.10 [OS_ObjectIdToInteger\(\)](#) static unsigned long OS_ObjectIdToInteger ([osal_id_t](#) *object_id*) [inline], [static]

Obtain an integer value corresponding to an object ID.

Obtains an integer representation of an object id, generally for the purpose of printing to the console or system logs. The returned value is of the type "unsigned long" for direct use with printf-style functions. It is recommended to use the "%lx" conversion specifier as the hexadecimal encoding clearly delineates the internal fields.

Note

This provides the raw integer value and is *not* suitable for use as an array index, as the result is not zero-based. See the [OS_ConvertToArrayIndex\(\)](#) to obtain a zero-based index value.

Parameters

in	<i>object_id</i>	The object ID
----	------------------	---------------

Returns

integer value representation of object ID

Definition at line 80 of file osapi-idmap.h.

Referenced by [CF_WrappedClose\(\)](#), [OS_ObjectIdDefined\(\)](#), and [OS_ObjectIdEqual\(\)](#).

10.71 OSAL Dynamic Loader and Symbol APIs

Functions

- `int32 OS_SymbolLookup (cpuaddr *symbol_address, const char *symbol_name)`
Find the Address of a Symbol.
- `int32 OS_ModuleSymbolLookup (osal_id_t module_id, cpuaddr *symbol_address, const char *symbol_name)`
Find the Address of a Symbol within a module.
- `int32 OS_SymbolTableDump (const char *filename, size_t size_limit)`
Dumps the system symbol table to a file.
- `int32 OS_ModuleLoad (osal_id_t *module_id, const char *module_name, const char *filename, uint32 flags)`
Loads an object file.
- `int32 OS_ModuleUnload (osal_id_t module_id)`
Unloads the module file.
- `int32 OS_ModuleInfo (osal_id_t module_id, OS_module_prop_t *module_info)`
Obtain information about a module.

10.71.1 Detailed Description

10.71.2 Function Documentation

10.71.2.1 OS_ModuleInfo() `int32 OS_ModuleInfo (`
 `osal_id_t module_id,`
 `OS_module_prop_t * module_info)`

Obtain information about a module.

Returns information about the loadable module

Parameters

in	<code>module_id</code>	OSAL ID of the previously the loaded module
out	<code>module_info</code>	Buffer to store module information (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the module id invalid
<code>OS_INVALID_POINTER</code>	if the pointer to the ModuleInfo structure is invalid
<code>OS_ERROR</code>	if an other/unspecified error occurs (return value only verified in coverage test)

10.71.2.2 OS_ModuleLoad() `int32 OS_ModuleLoad (`
 `osal_id_t * module_id,`
 `const char * module_name,`
 `const char * filename,`

```
    uint32 flags )
```

Loads an object file.

Loads an object file into the running operating system

The "flags" parameter may influence how the loaded module symbols are made available for use in the application. See [OS_MODULE_FLAG_LOCAL_SYMBOLS](#) and [OS_MODULE_FLAG_GLOBAL_SYMBOLS](#) for descriptions.

Parameters

out	<i>module_id</i>	Non-zero OSAL ID corresponding to the loaded module
in	<i>module_name</i>	Name of module (must not be null)
in	<i>filename</i>	File containing the object code to load (must not be null)
in	<i>flags</i>	Options for the loaded module

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
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
OS_ERR_NAME_TOO_LONG	if the module_name is too long
OS_FS_ERR_PATH_INVALID	if the filename argument is not valid
OS_ERROR	if an other/unspecified error occurs (return value only verified in coverage test)

10.71.2.3 OS_ModuleSymbolLookup() [int32 OS_ModuleSymbolLookup \(](#)

```
    osal_id_t module_id,
    cpuaddr * symbol_address,
    const char * symbol_name )
```

Find the Address of a Symbol within a module.

This is similar to [OS_SymbolLookup\(\)](#) but for a specific module ID. This should be used to look up a symbol in a module that has been loaded with the [OS_MODULE_FLAG_LOCAL_SYMBOLS](#) flag.

Parameters

in	<i>module_id</i>	Module ID that should contain the symbol
out	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
----------------------------	-----------------------

Return values

<i>OS_ERROR</i>	if the symbol could not be found
<i>OS_INVALID_POINTER</i>	if one of the pointers passed in are NULL

10.71.2.4 OS_ModuleUnload() `int32 OS_ModuleUnload (osal_id_t module_id)`

Unloads the module file.

Unloads the module file from the running operating system

Parameters

in	<i>module_id</i>	OSAL ID of the previously the loaded module
----	------------------	---

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the module id invalid
<i>OS_ERROR</i>	if an other/unspecified error occurs (return value only verified in coverage test)

10.71.2.5 OS_SymbolLookup() `int32 OS_SymbolLookup (cpuaddr * symbol_address, const char * symbol_name)`

Find the Address of a Symbol.

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	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the symbol could not be found
<i>OS_INVALID_POINTER</i>	if one of the pointers passed in are NULL

```
10.71.2.6 OS_SymbolTableDump() int32 OS_SymbolTableDump (
    const char * filename,
    size_t size_limit )
```

Dumps the system symbol table to a file.

Dumps the system symbol table to the specified filename

Note

Not all RTOS implementations support this API. If the underlying module subsystem does not provide a facility to iterate through the symbol table, then the [OS_ERR_NOT_IMPLEMENTED](#) status code is returned.

Parameters

in	<i>filename</i>	File to write to (must not be null)
in	<i>size_limit</i>	Maximum number of bytes to write

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_NOT_IMPLEMENTED	Not implemented.
OS_INVALID_POINTER	if the filename argument is NULL
OS_FS_ERR_PATH_INVALID	if the filename argument is not valid
OS_ERR_NAME_TOO_LONG	if any of the symbol names are too long (return value only verified in coverage test)
OS_ERR_OUTPUT_TOO_LARGE	if the size_limit was reached before completing all symbols (return value only verified in coverage test)
OS_ERROR	if an other/unspecified error occurs (return value only verified in coverage test)

10.72 OSAL Mutex APIs

Functions

- `int32 OS_MutSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 options)`
Creates a mutex semaphore.
- `int32 OS_MutSemGive (osal_id_t sem_id)`
Releases the mutex object referenced by sem_id.
- `int32 OS_MutSemTake (osal_id_t sem_id)`
Acquire the mutex object referenced by sem_id.
- `int32 OS_MutSemDelete (osal_id_t sem_id)`
Deletes the specified Mutex Semaphore.
- `int32 OS_MutSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`
Find an existing mutex ID by name.
- `int32 OS_MutSemGetInfo (osal_id_t sem_id, OS_mut_sem_prop_t *mut_prop)`
Fill a property object buffer with details regarding the resource.

10.72.1 Detailed Description

10.72.2 Function Documentation

10.72.2.1 OS_MutSemCreate() `int32 OS_MutSemCreate (`

```
    osal_id_t * sem_id,
    const char * sem_name,
    uint32 options )
```

Creates a mutex semaphore.

Mutex semaphores are always created in the unlocked (full) state.

Parameters

out	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>sem_name</code>	the name of the new resource to create (must not be null)
in	<code>options</code>	reserved for future use. Should be passed as 0.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if sem_id or sem_name are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NO_FREE_IDS</code>	if there are no more free mutex IDs
<code>OS_ERR_NAME_TAKEN</code>	if there is already a mutex with the same name
<code>OS_SEM_FAILURE</code>	if the OS call failed (return value only verified in coverage test)

10.72.2.2 OS_MutSemDelete() `int32 OS_MutSemDelete (osal_id_t sem_id)`

Deletes the specified Mutex Semaphore.

Delete the semaphore. This also frees the respective sem_id such that it can be used again when another is created.

Parameters

in	<code>sem_id</code>	The object ID to delete
----	---------------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid mutex
<code>OS_SEM_FAILURE</code>	if an unspecified error occurs (return value only verified in coverage test)

10.72.2.3 OS_MutSemGetIdByName() `int32 OS_MutSemGetIdByName (`

```
    osal_id_t * sem_id,
    const char * sem_name )
```

Find an existing mutex ID by 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	<code>sem_id</code>	will be set to the ID of the existing resource
in	<code>sem_name</code>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

10.72.2.4 OS_MutSemGetInfo() `int32 OS_MutSemGetInfo (`

```
    osal_id_t sem_id,
    OS_mut_sem_prop_t * mut_prop )
```

Fill a property object buffer with details regarding the resource.

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	<i>sem_id</i>	The object ID to operate on
out	<i>mut_prop</i>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the mut_prop pointer is null

10.72.2.5 OS_MutSemGive() `int32 OS_MutSemGive (osal_id_t sem_id)`

Releases the mutex object referenced by *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	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid mutex
<i>OS_SEM_FAILURE</i>	if an unspecified error occurs (return value only verified in coverage test)

10.72.2.6 OS_MutSemTake() `int32 OS_MutSemTake (osal_id_t sem_id)`

Acquire the mutex object referenced by *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	<i>sem->_id</i>	The object ID to operate on
----	--------------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the id passed in is not a valid mutex
<i>OS_SEM_FAILURE</i>	if an unspecified error occurs (return value only verified in coverage test)

10.73 OSAL Network ID APIs

Provides some basic methods to query a network host name and ID.

Functions

- `int32 OS_NetworkGetID (void)`
Gets the network ID of the local machine.
- `int32 OS_NetworkGetHostName (char *host_name, size_t name_len)`
Gets the local machine network host name.

10.73.1 Detailed Description

Provides some basic methods to query a network host name and ID.

10.73.2 Function Documentation

10.73.2.1 OS_NetworkGetHostName() `int32 OS_NetworkGetHostName (`
 `char * host_name,`
 `size_t name_len)`

Gets the local machine network host name.

If configured in the underlying network stack, this function retrieves the local hostname of the system.

Parameters

<code>out</code>	<code>host_name</code>	Buffer to hold name information (must not be null)
<code>in</code>	<code>name_len</code>	Maximum length of host name buffer (must not be zero)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_SIZE</code>	if the name_len is zero
<code>OS_INVALID_POINTER</code>	if the host_name is NULL

10.73.2.2 OS_NetworkGetID() `int32 OS_NetworkGetID (`
 `void)`

Gets the network ID of the local machine.

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

The 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.

10.74 OSAL Printf APIs

Functions

- void [OS_printf](#) (const char *string,...) [OS_PRINTF](#)(
Abstraction for the system printf() call.)
- void void [OS_printf_disable](#) (void)
This function disables the output from OS_printf.
- void [OS_printf_enable](#) (void)
This function enables the output from OS_printf.

10.74.1 Detailed Description

10.74.2 Function Documentation

10.74.2.1 OS_printf() void OS_printf (
 const char * string,
 ...)

Abstraction for the system printf() call.

This function abstracts out the printf type statements. This is useful for using OS- specific thots 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 and takes all the parameters and formatting options of printf. This abstraction may implement additional buffering, if necessary, to improve the real-time performance of the call.

Strings (including terminator) longer than [OS_BUFFER_SIZE](#) will be truncated.

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	<i>string</i>	Format string, followed by additional arguments
----	---------------	---

10.74.2.2 OS_printf_disable() void void OS_printf_disable (
 void)

This function disables the output from OS_printf.

10.74.2.3 OS_printf_enable() void OS_printf_enable (
 void)

This function enables the output from OS_printf.

10.75 OSAL Message Queue APIs

Functions

- `int32 OS_QueueCreate (osal_id_t *queue_id, const char *queue_name, osal_blockcount_t queue_depth, size_t data_size, uint32 flags)`

Create a message queue.
- `int32 OS_QueueDelete (osal_id_t queue_id)`

Deletes the specified message queue.
- `int32 OS_QueueGet (osal_id_t queue_id, void *data, size_t size, size_t *size_copied, int32 timeout)`

Receive a message on a message queue.
- `int32 OS_QueuePut (osal_id_t queue_id, const void *data, size_t size, uint32 flags)`

Put a message on a message queue.
- `int32 OS_QueueGetIdByName (osal_id_t *queue_id, const char *queue_name)`

Find an existing queue ID by name.
- `int32 OS_QueueGetInfo (osal_id_t queue_id, OS_queue_prop_t *queue_prop)`

Fill a property object buffer with details regarding the resource.

10.75.1 Detailed Description

10.75.2 Function Documentation

10.75.2.1 OS_QueueCreate() `int32 OS_QueueCreate (`
`osal_id_t * queue_id,`
`const char * queue_name,`
`osal_blockcount_t queue_depth,`
`size_t data_size,`
`uint32 flags)`

Create a message queue.

This is the function used to create a queue in the operating system. Depending on the underlying operating system, the memory for the queue will be allocated automatically or allocated by the code that sets up the queue. Queue names must be unique; if the name already exists this function fails. Names cannot be NULL.

Parameters

<code>out</code>	<code>queue_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>queue_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>queue_depth</code>	the maximum depth of the queue
<code>in</code>	<code>data_size</code>	the size of each entry in the queue (must not be zero)
<code>in</code>	<code>flags</code>	options for the queue (reserved for future use, pass as 0)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if a pointer passed in is NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>

Return values

<i>OS_ERR_NO_FREE_IDS</i>	if there are already the max queues created
<i>OS_ERR_NAME_TAKEN</i>	if the name is already being used on another queue
<i>OS_ERR_INVALID_SIZE</i>	if data_size is 0
<i>OS_QUEUE_INVALID_SIZE</i>	if the queue depth exceeds the limit
<i>OS_ERROR</i>	if the OS create call fails

```
10.75.2.2 OS_QueueDelete() int32 OS_QueueDelete (
    osal_id_t queue_id )
```

Deletes the specified message queue.

This is the function used to delete a queue in the operating system. This also frees the respective queue_id to be used again when another queue is created.

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	<i>queue_id</i>	The object ID to delete
----	-----------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in does not exist
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

```
10.75.2.3 OS_QueueGet() int32 OS_QueueGet (
```

```
    osal_id_t queue_id,
    void * data,
    size_t size,
    size_t * size_copied,
    int32 timeout )
```

Receive a message on a message queue.

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	<i>queue_id</i>	The object ID to operate on
----	-----------------	-----------------------------

Parameters

<i>out</i>	<i>data</i>	The buffer to store the received message (must not be null)
<i>in</i>	<i>size</i>	The size of the data buffer (must not be zero)
<i>out</i>	<i>size_copied</i>	Set to the actual size of the message (must not be null)
<i>in</i>	<i>timeout</i>	The maximum amount of time to block, or OS_PEND to wait forever

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the given ID does not exist
<i>OS_INVALID_POINTER</i>	if a pointer passed in is NULL
<i>OS_QUEUE_EMPTY</i>	if the Queue has no messages on it to be received
<i>OS_QUEUE_TIMEOUT</i>	if the timeout was OS_PEND and the time expired
<i>OS_QUEUE_INVALID_SIZE</i>	if the size copied from the queue was not correct
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

10.75.2.4 OS_QueueGetIdByName() `int32 OS_QueueGetIdByName (`

```
    osal_id_t * queue_id,
    const char * queue_name )
```

Find an existing queue ID by 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

<i>out</i>	<i>queue_id</i>	will be set to the ID of the existing resource
<i>in</i>	<i>queue_name</i>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if the name or id pointers are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	the name was not found in the table

10.75.2.5 OS_QueueGetInfo() `int32 OS_QueueGetInfo (`

```
    osal_id_t queue_id,  
    OS_QUEUE_PROP_T * queue_prop )
```

Fill a property object buffer with details regarding the resource.

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	<i>queue_id</i>	The object ID to operate on
out	<i>queue_prop</i>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if queue_prop is NULL
<i>OS_ERR_INVALID_ID</i>	if the ID given is not a valid queue

10.75.2.6 OS_QueuePut() `int32 OS_QueuePut(`

```
    osal_id_t queue_id,  
    const void * data,  
    size_t size,  
    uint32 flags )
```

Put a message on a message queue.

Parameters

in	<i>queue_id</i>	The object ID to operate on
in	<i>data</i>	The buffer containing the message to put (must not be null)
in	<i>size</i>	The size of the data buffer (must not be zero)
in	<i>flags</i>	Currently reserved/unused, should be passed as 0

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the queue id passed in is not a valid queue
<i>OS_INVALID_POINTER</i>	if the data pointer is NULL
<i>OS_QUEUE_INVALID_SIZE</i>	if the data message is too large for the queue
<i>OS_QUEUE_FULL</i>	if the queue cannot accept another message
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

10.76 OSAL Select APIs

Functions

- `int32 OS_SelectMultipleAbs (OS_FdSet *ReadSet, OS_FdSet *WriteSet, OS_time_t abs_timeout)`
Wait for events across multiple file handles.
- `int32 OS_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)`
Wait for events across multiple file handles.
- `int32 OS_SelectSingleAbs (osal_id_t objid, uint32 *StateFlags, OS_time_t abs_timeout)`
Wait for events on a single file handle.
- `int32 OS_SelectSingle (osal_id_t 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, osal_id_t objid)`
Add an ID to an FdSet structure.
- `int32 OS_SelectFdClear (OS_FdSet *Set, osal_id_t objid)`
Clear an ID from an FdSet structure.
- `bool OS_SelectFdsSet (const OS_FdSet *Set, osal_id_t objid)`
Check if an FdSet structure contains a given ID.

10.76.1 Detailed Description

10.76.2 Function Documentation

10.76.2.1 OS_SelectFdAdd() `int32 OS_SelectFdAdd (`
`OS_FdSet * Set,`
`osal_id_t objid)`

Add an ID to an FdSet structure.

After this call the set will contain the given OSAL ID

Parameters

<code>in, out</code>	<code>Set</code>	Pointer to <code>OS_FdSet</code> object to operate on (must not be null)
<code>in</code>	<code>objid</code>	The handle ID to add to the set

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if argument is NULL
<code>OS_ERR_INVALID_ID</code>	if the objid is not a valid handle

10.76.2.2 OS_SelectFdClear() `int32 OS_SelectFdClear (`

```
OS_FdSet * Set,
osal_id_t objid )
```

Clear an ID from an FdSet structure.

After this call the set will no longer contain the given OSAL ID

Parameters

in, out	<i>Set</i>	Pointer to OS_FdSet object to operate on (must not be null)
in	<i>objid</i>	The handle ID to remove from the set

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_ID	if the objid is not a valid handle

```
10.76.2.3 OS_SelectFdIsSet() bool OS_SelectFdIsSet (
    const OS_FdSet * Set,
    osal_id_t objid )
```

Check if an FdSet structure contains a given ID.

Parameters

in	<i>Set</i>	Pointer to OS_FdSet object to operate on (must not be null)
in	<i>objid</i>	The handle ID to check for in the set

Returns

Boolean set status

Return values

<i>true</i>	FdSet structure contains ID
<i>false</i>	FdSet structure does not contain ID

```
10.76.2.4 OS_SelectFdZero() int32 OS_SelectFdZero (
    OS_FdSet * Set )
```

Clear a FdSet structure.

After this call the set will contain no OSAL IDs

Parameters

out	<i>Set</i>	Pointer to OS_FdSet object to clear (must not be null)
-----	------------	--

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if argument is NULL

10.76.2.5 OS_SelectMultiple() `int32 OS_SelectMultiple (`

```
    OS_FdSet * ReadSet,
    OS_FdSet * WriteSet,
    int32 msecs )
```

Wait for events across multiple file handles.

Wait for any of the given sets of IDs to 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.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS_SelectMultipleAbs\(\)](#) for higher timing precision.

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.

Parameters

<code>in, out</code>	<code>ReadSet</code>	Set of handles to check/wait to become readable
<code>in, out</code>	<code>WriteSet</code>	Set of handles to check/wait to become writable
<code>in</code>	<code>msecs</code>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

See also

[OS_SelectMultipleAbs\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	If any handle in the ReadSet or WriteSet is readable or writable, respectively
<code>OS_ERROR_TIMEOUT</code>	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
<code>OS_ERR_OPERATION_NOT_SUPPORTED</code>	if a specified handle does not support select
<code>OS_ERR_INVALID_ID</code>	if no valid handles were contained in the ReadSet/WriteSet

10.76.2.6 OS_SelectMultipleAbs()

```
int32 OS_SelectMultipleAbs (
    OS_FdSet * ReadSet,
    OS_FdSet * WriteSet,
    OS_time_t abs_timeout )
```

Wait for events across multiple file handles.

Wait for any of the given sets of IDs to 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.

This API is identical to [OS_SelectMultiple\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS_SelectMultiple\(\)](#).

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.

Parameters

<code>in, out</code>	<code>ReadSet</code>	Set of handles to check/wait to become readable
<code>in, out</code>	<code>WriteSet</code>	Set of handles to check/wait to become writable
<code>in</code>	<code>abs_timeout</code>	The absolute time that the call may block until

See also

[OS_SelectMultiple\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	If any handle in the ReadSet or WriteSet is readable or writable, respectively
<i>OS_ERROR_TIMEOUT</i>	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
<i>OS_ERR_OPERATION_NOT_SUPPORTED</i>	if a specified handle does not support select
<i>OS_ERR_INVALID_ID</i>	if no valid handles were contained in the ReadSet/WriteSet

```
10.76.2.7 OS_SelectSingle() int32 OS_SelectSingle (
    osal_id_t objid,
    uint32 * StateFlags,
    int32 msecs )
```

Wait for events on a single file handle.

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 "StateFlags" parameter should be set to the desired state (OS_STREAM_STATE_READABLE and/or OS_STREAM_STATE_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.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS_SelectSingleAbs\(\)](#) for higher timing precision.

Parameters

in	<i>objid</i>	The handle ID to select on
in, out	<i>StateFlags</i>	State flag(s) (readable or writable) (must not be null)
in	<i>msecs</i>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

See also

[OS_SelectSingleAbs\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	If the handle is readable and/or writable, as requested
<i>OS_ERROR_TIMEOUT</i>	If the handle did not become readable or writable within the timeout
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the objid is not a valid handle

```
10.76.2.8 OS_SelectSingleAbs() int32 OS_SelectSingleAbs (
    osal_id_t objid,
    uint32 * StateFlags,
    OS_time_t abs_timeout )
```

Wait for events on a single file handle.

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 "StateFlags" parameter should be set to the desired state (OS_STREAM_STATE_READABLE and/or OS_STREAM_STATE_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.

This API is identical to [OS_SelectSingle\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS_SelectSingle\(\)](#).

Parameters

in	<i>objid</i>	The handle ID to select on
in, out	<i>StateFlags</i>	State flag(s) (readable or writable) (must not be null)
in	<i>abs_timeout</i>	The absolute time that the call may block until

See also

[OS_SelectSingle\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	If the handle is readable and/or writable, as requested
OS_ERROR_TIMEOUT	If the handle did not become readable or writable within the timeout
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_ID	if the objid is not a valid handle

10.77 OSAL Shell APIs

Functions

- int32 `OS_ShellOutputToFile` (const char *Cmd, `osal_id_t` filedes)

Executes the command and sends output to a file.

10.77.1 Detailed Description

10.77.2 Function Documentation

10.77.2.1 `OS_ShellOutputToFile()`

```
int32 OS_ShellOutputToFile (
    const char * Cmd,
    osal_id_t filedes )
```

Executes the command and sends output to a file.

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	<i>Cmd</i>	Command to pass to shell (must not be null)
in	<i>filedes</i>	File to send output to.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERROR</code>	if the command was not executed properly
<code>OS_INVALID_POINTER</code>	if Cmd argument is NULL
<code>OS_ERR_INVALID_ID</code>	if the file descriptor passed in is invalid

10.78 OSAL Socket Address APIs

These functions provide a means to manipulate network addresses in a manner that is (mostly) agnostic to the actual network address type.

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, size_t 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.

10.78.1 Detailed Description

These functions provide a means to manipulate network addresses in a manner that is (mostly) agnostic to the actual network address type.

Every network address should be representable as a string (i.e. dotted decimal IP, etc). This can serve as the "common denominator" to all address types.

10.78.2 Function Documentation

10.78.2.1 OS_SocketAddrFromString() `int32 OS_SocketAddrFromString (` `OS_SockAddr_t * Addr,` `const char * string)`

Set a network host address from a string representation.

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

<code>out</code>	<code>Addr</code>	The address buffer to initialize (must not be null)
<code>in</code>	<code>string</code>	The string to initialize the address from (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERROR</i>	if the string cannot be converted to an address

10.78.2.2 OS_SocketAddrGetPort() `int32 OS_SocketAddrGetPort (`

```
    uint16 * PortNum,
    const OS_SockAddr_t * Addr )
```

Get the port number of a network address.

For network protocols 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

<i>out</i>	<i>PortNum</i>	Buffer to store the port number (must not be null)
<i>in</i>	<i>Addr</i>	The network address buffer (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_BAD_ADDRESS</i>	if the address domain is not compatible

10.78.2.3 OS_SocketAddrInit() `int32 OS_SocketAddrInit (`

```
    OS_SockAddr_t * Addr,
    OS_SocketDomain_t Domain )
```

Initialize a socket address structure to hold an address of the given family.

The address is set to a suitable default value for the family.

Parameters

<i>out</i>	<i>Addr</i>	The address buffer to initialize (must not be null)
<i>in</i>	<i>Domain</i>	The address family

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
-------------------	-----------------------

Return values

<i>OS_INVALID_POINTER</i>	if Addr argument is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	if the system does not implement the requested domain

10.78.2.4 OS_SocketAddrSetPort() `int32 OS_SocketAddrSetPort (`

```
    OS_SockAddr_t * Addr,  
    uint16 PortNum )
```

Set the port number of a network address.

For network protocols 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

<i>out</i>	<i>Addr</i>	The network address buffer (must not be null)
<i>in</i>	<i>PortNum</i>	The port number to set

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_BAD_ADDRESS</i>	if the address domain is not compatible

10.78.2.5 OS_SocketAddrToString() `int32 OS_SocketAddrToString (`

```
    char * buffer,  
    size_t bufLen,  
    const OS_SockAddr_t * Addr )
```

Get a string representation of a network host address.

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

<i>out</i>	<i>buffer</i>	Buffer to hold the output string (must not be null)
<i>in</i>	<i>bufLen</i>	Maximum length of the output string (must not be zero)
<i>in</i>	<i>Addr</i>	The network address buffer to convert (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERROR</i>	if the address cannot be converted to string, or string buffer too small

10.79 OSAL Socket Management APIs

These functions are loosely related to the BSD Sockets API but made to be more consistent with other OSAL API functions. That is, they operate on OSAL IDs (32-bit opaque number values) and return an OSAL error code.

Functions

- `int32 OS_SocketOpen (osal_id_t *sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)`
Opens a socket.
- `int32 OS_SocketBind (osal_id_t sock_id, const OS_SockAddr_t *Addr)`
Binds a socket to a given local address and enter listening (server) mode.
- `int32 OS_SocketListen (osal_id_t sock_id)`
Places the specified socket into a listening state.
- `int32 OS_SocketBindAddress (osal_id_t sock_id, const OS_SockAddr_t *Addr)`
Binds a socket to a given local address.
- `int32 OS_SocketConnectAbs (osal_id_t sock_id, const OS_SockAddr_t *Addr, OS_time_t abs_timeout)`
Connects a socket to a given remote address.
- `int32 OS_SocketConnect (osal_id_t sock_id, const OS_SockAddr_t *Addr, int32 timeout)`
Connects a socket to a given remote address.
- `int32 OS_SocketShutdown (osal_id_t sock_id, OS_SocketShutdownMode_t Mode)`
Implement graceful shutdown of a stream socket.
- `int32 OS_SocketAcceptAbs (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, OS_time_t abs_timeout)`
Waits for and accept the next incoming connection on the given socket.
- `int32 OS_SocketAccept (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, int32 timeout)`
Waits for and accept the next incoming connection on the given socket.
- `int32 OS_SocketRecvFromAbs (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, OS_time_t abs_timeout)`
Reads data from a message-oriented (datagram) socket.
- `int32 OS_SocketRecvFrom (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, int32 timeout)`
Reads data from a message-oriented (datagram) socket.
- `int32 OS_SocketSendTo (osal_id_t sock_id, const void *buffer, size_t buflen, const OS_SockAddr_t *RemoteAddr)`
Sends data to a message-oriented (datagram) socket.
- `int32 OS_SocketGetIdByName (osal_id_t *sock_id, const char *sock_name)`
Gets an OSAL ID from a given name.
- `int32 OS_SocketGetInfo (osal_id_t sock_id, OS_socket_prop_t *sock_prop)`
Gets information about an OSAL Socket ID.

10.79.1 Detailed Description

These functions are loosely related to the BSD Sockets API but made to be more consistent with other OSAL API functions. That is, they operate on OSAL IDs (32-bit opaque number values) and return an OSAL error code. OSAL Socket IDs are very closely related to File IDs and share the same ID number space. Additionally, the file `OS_read()` / `OS_write()` / `OS_close()` calls also work on sockets.

Note that all of functions may return `OS_ERR_NOT_IMPLEMENTED` if network support is not configured at compile time.

10.79.2 Function Documentation

```
10.79.2.1 OS_SocketAccept() int32 OS_SocketAccept (
    osal_id_t sock_id,
    osal_id_t * connsock_id,
    OS_SockAddr_t * Addr,
    int32 timeout )
```

Waits for and accept the next incoming connection on the given socket.

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.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS_SocketAcceptAbs\(\)](#) for higher timing precision.

Parameters

in	<i>sock_id</i>	The server socket ID, previously bound using OS_SocketBind()
out	<i>connsock_id</i>	The connection socket, a new ID that can be read/written (must not be null)
in	<i>Addr</i>	The remote address of the incoming connection (must not be null)
in	<i>timeout</i>	The maximum amount of time to wait, or OS_PEND to wait forever

See also

[OS_SocketAcceptAbs\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_ID	if the <i>sock_id</i> parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket
OS_ERR_INCORRECT_OBJ_STATE	if the socket is not bound or already connected

```
10.79.2.2 OS_SocketAcceptAbs() int32 OS_SocketAcceptAbs (
    osal_id_t sock_id,
    osal_id_t * connsock_id,
    OS_SockAddr_t * Addr,
    OS_time_t abs_timeout )
```

Waits for and accept the next incoming connection on the given socket.

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.

This API is identical to [OS_SocketAccept\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS_SocketAccept\(\)](#).

Parameters

in	<i>sock_id</i>	The server socket ID, previously bound using OS_SocketBind()
out	<i>connsock_id</i>	The connection socket, a new ID that can be read/written (must not be null)
in	<i>Addr</i>	The remote address of the incoming connection (must not be null)
in	<i>abs_timeout</i>	The absolute time that the call may block until

See also

[OS_SocketAccept\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_ID	if the <i>sock_id</i> parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket
OS_ERR_INCORRECT_OBJ_STATE	if the socket is not bound or already connected

```
10.79.2.3 OS_SocketBind() int32 OS_SocketBind (
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr )
```

Binds a socket to a given local address and enter listening (server) mode.

This is a convenience/compatibility routine to perform both [OS_SocketBindAddress\(\)](#) and [OS_SocketListen\(\)](#) operations in a single call, intended to simplify the setup for a server role.

If the socket is connectionless, then it only binds to the local address.

Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
----------------------------	-----------------------

Return values

<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already bound
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.79.2.4 OS_SocketBindAddress() int32 OS_SocketBindAddress (
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr )
```

Binds a socket to a given local address.

The specified socket will be bound to the local address and port, if available. This controls the source address reflected in network traffic transmitted via this socket.

After binding to the address, a stream socket may be followed by a call to either [OS_SocketListen\(\)](#) for a server role or to [OS_SocketConnect\(\)](#) for a client role.

Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already bound
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.79.2.5 OS_SocketConnect() int32 OS_SocketConnect (
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr,
    int32 timeout )
```

Connects a socket to a given remote address.

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).

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS_SocketConnectAbs\(\)](#) for higher timing precision.

Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The remote address to connect to (must not be null)
in	<i>timeout</i>	The maximum amount of time to wait, or OS_PEND to wait forever

See also

[OS_SocketConnectAbs\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already connected
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_INVALID_POINTER</i>	if <i>Addr</i> argument is NULL

10.79.2.6 OS_SocketConnectAbs() `int32 OS_SocketConnectAbs (`

```
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr,
    OS_time_t abs_timeout )
```

Connects a socket to a given remote address.

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).

This API is identical to [OS_SocketConnect\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS_SocketConnect\(\)](#).

Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The remote address to connect to (must not be null)
in	<i>abs_timeout</i>	The absolute time that the call may block until

See also

[OS_SocketConnect\(\)](#)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already connected
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_INVALID_POINTER</i>	if <i>Addr</i> argument is NULL

10.79.2.7 OS_SocketGetIdByName() `int32 OS_SocketGetIdByName (`

```
    osal_id_t * sock_id,
    const char * sock_name )
```

Gets an OSAL ID from a given name.

Note

OSAL Sockets use generated names according to the address and type.

See also

[OS_SocketGetInfo\(\)](#)

Parameters

<i>out</i>	<i>sock_id</i>	Buffer to hold result (must not be null)
<i>in</i>	<i>sock_name</i>	Name of socket to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	is id or name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

10.79.2.8 OS_SocketGetInfo() `int32 OS_SocketGetInfo (`

```
    osal_id_t sock_id,
    OS_socket_prop_t * sock_prop )
```

Gets information about an OSAL Socket ID.

OSAL Sockets use generated names according to the address and type. This allows applications to find the name of a given socket.

Parameters

<i>in</i>	<i>sock_id</i>	The socket ID
-----------	----------------	---------------

Parameters

out	<i>sock_prop</i>	Buffer to hold socket information (must not be null)
-----	------------------	--

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the count_prop pointer is null

10.79.2.9 OS_SocketListen() `int32 OS_SocketListen (osal_id_t sock_id)`

Places the specified socket into a listening state.

This function only applies to connection-oriented (stream) sockets that are intended to be used in a server-side role. This places the socket into a state where it can accept incoming connections from clients.

Parameters

in	<i>sock_id</i>	The socket ID
----	----------------	---------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the sock_id parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already listening
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a stream socket

10.79.2.10 OS_SocketOpen() `int32 OS_SocketOpen (osal_id_t * sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)`

Opens a socket.

A new, unconnected and unbound socket is allocated of the given domain and type.

Parameters

out	<i>sock_id</i>	Buffer to hold the non-zero OSAL ID (must not be null)
-----	----------------	--

Parameters

in	<i>Domain</i>	The domain / address family of the socket (INET or INET6, etc)
in	<i>Type</i>	The type of the socket (STREAM or DATAGRAM)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	if the system does not implement the requested socket/address domain

10.79.2.11 OS_SocketRecvFrom() `int32 OS_SocketRecvFrom (`

```
    osal_id_t sock_id,
    void * buffer,
    size_t buflen,
    OS_SockAddr_t * RemoteAddr,
    int32 timeout )
```

Reads data from a message-oriented (datagram) socket.

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.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS_SocketRecvFromAbs\(\)](#) for higher timing precision.

Parameters

in	<i>sock_id</i>	The socket ID, previously bound using OS_SocketBind()
out	<i>buffer</i>	Pointer to message data receive buffer (must not be null)
in	<i>buflen</i>	The maximum length of the message data to receive (must not be zero)
out	<i>RemoteAddr</i>	Buffer to store the remote network address (may be NULL)
in	<i>timeout</i>	The maximum amount of time to wait or OS_PEND to wait forever

See also

[OS_SocketRecvFromAbs\(\)](#)

Returns

Count of actual bytes received or error status, see [OSAL Return Code Defines](#)

Return values

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERR_INVALID_ID</i>	if the sock_id parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

10.79.2.12 OS_SocketRecvFromAbs() `int32 OS_SocketRecvFromAbs (`

```
    osal_id_t sock_id,
    void * buffer,
    size_t buflen,
    OS_SockAddr_t * RemoteAddr,
    OS_time_t abs_timeout )
```

Reads data from a message-oriented (datagram) socket.

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.

This API is identical to [OS_SocketRecvFrom\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS_SocketRecvFrom\(\)](#).

Parameters

in	<i>sock_id</i>	The socket ID, previously bound using OS_SocketBind()
out	<i>buffer</i>	Pointer to message data receive buffer (must not be null)
in	<i>buflen</i>	The maximum length of the message data to receive (must not be zero)
out	<i>RemoteAddr</i>	Buffer to store the remote network address (may be NULL)
in	<i>abs_timeout</i>	The absolute time at which the call should return if nothing received

Returns

Count of actual bytes received or error status, see [OSAL Return Code Defines](#)

Return values

OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_SIZE	if passed-in buflen is not valid
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket

10.79.2.13 OS_SocketSendTo() `int32 OS_SocketSendTo (`

```
    osal_id_t sock_id,
    const void * buffer,
    size_t buflen,
    const OS_SockAddr_t * RemoteAddr )
```

Sends data to a message-oriented (datagram) socket.

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	<i>sock_id</i>	The socket ID, which must be of the datagram type
in	<i>buffer</i>	Pointer to message data to send (must not be null)
in	<i>buflen</i>	The length of the message data to send (must not be zero)
in	<i>RemoteAddr</i>	Buffer containing the remote network address to send to

Returns

Count of actual bytes sent or error status, see [OSAL Return Code Defines](#)

Return values

<code>OS_INVALID_POINTER</code>	if argument is NULL
<code>OS_ERR_INVALID_SIZE</code>	if passed-in buflen is not valid
<code>OS_ERR_INVALID_ID</code>	if the sock_id parameter is not valid
<code>OS_ERR_INCORRECT_OBJ_TYPE</code>	if the handle is not a socket

10.79.2.14 OS_SocketShutdown() `int32 OS_SocketShutdown (`

```
    osal_id_t sock_id,
    OS_SocketShutdownMode_t Mode )
```

Implement graceful shutdown of a stream socket.

This can be utilized to indicate the end of data stream without immediately closing the socket, giving the remote side an indication that the data transfer is complete.

Parameters

in	<code>sock_id</code>	The socket ID
in	<code>Mode</code>	Whether to shutdown reading, writing, or both.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the sock_id parameter is not valid
<code>OS_ERR_INVALID_ARGUMENT</code>	if the Mode argument is not one of the valid options
<code>OS_ERR_INCORRECT_OBJ_TYPE</code>	if the handle is not a socket
<code>OS_ERR_INCORRECT_OBJ_STATE</code>	if the socket is not connected

10.80 OSAL Task APIs

Functions

- `int32 OS_TaskCreate (osal_id_t *task_id, const char *task_name, osal_task_entry function_pointer, osal_stackptr_t stack_pointer, size_t stack_size, osal_priority_t priority, uint32 flags)`
Creates a task and starts running it.
- `int32 OS_TaskDelete (osal_id_t 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 (osal_id_t task_id, osal_priority_t new_priority)`
Sets the given task to a new priority.
- `osal_id_t OS_TaskGetId (void)`
Obtain the task id of the calling task.
- `int32 OS_TaskGetIdByName (osal_id_t *task_id, const char *task_name)`
Find an existing task ID by name.
- `int32 OS_TaskGetInfo (osal_id_t task_id, OS_task_prop_t *task_prop)`
Fill a property object buffer with details regarding the resource.
- `int32 OS_TaskFindIdBySystemData (osal_id_t *task_id, const void *sysdata, size_t sysdata_size)`
Reverse-lookup the OSAL task ID from an operating system ID.

10.80.1 Detailed Description

10.80.2 Function Documentation

10.80.2.1 OS_TaskCreate() `int32 OS_TaskCreate (`
 `osal_id_t * task_id,`
 `const char * task_name,`
 `osal_task_entry function_pointer,`
 `osal_stackptr_t stack_pointer,`
 `size_t stack_size,`
 `osal_priority_t priority,`
 `uint32 flags)`

Creates a task and starts running it.

Creates a task and passes back the id of the task created. Task names must be unique; if the name already exists this function fails. Names cannot be NULL.

Portable applications should always specify the actual stack size in the stack_size parameter, not 0. This size value is not enforced/checked by OSAL, but is simply passed through to the RTOS for stack creation. Some RTOS implementations may assume 0 means a default stack size while others may actually create a task with no stack.

Unlike stack_size, the stack_pointer is optional and can be specified as NULL. In that case, a stack of the requested size will be dynamically allocated from the system heap.

Parameters

<code>out</code>	<code>task_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
------------------	----------------------	---

Parameters

in	<i>task_name</i>	the name of the new resource to create (must not be null)
in	<i>function_pointer</i>	the entry point of the new task (must not be null)
in	<i>stack_pointer</i>	pointer to the stack for the task, or NULL to allocate a stack from the system memory heap
in	<i>stack_size</i>	the size of the stack (must not be zero)
in	<i>priority</i>	initial priority of the new task
in	<i>flags</i>	initial options for the new task

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if any of the necessary pointers are NULL
<i>OS_ERR_INVALID_SIZE</i>	if the <i>stack_size</i> argument is zero
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_INVALID_PRIORITY</i>	if the priority is bad (return value only verified in coverage test)
<i>OS_ERR_NO_FREE_IDS</i>	if there can be no more tasks created
<i>OS_ERR_NAME_TAKEN</i>	if the name specified is already used by a task
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

10.80.2.2 OS_TaskDelay() `int32 OS_TaskDelay (uint32 millisecond)`

Delay a task for specified amount of milliseconds.

Causes the current thread to be suspended from execution for the period of millisecond. This is a scheduled wait (clock_nanosleep/rtems_task_wake_after/taskDelay), not a "busy" wait.

Parameters

in	<i>millisecond</i>	Amount of time to delay
----	--------------------	-------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

Referenced by CF_CFDP_InitEngine().

10.80.2.3 OS_TaskDelete() `int32 OS_TaskDelete (`
 `osal_id_t task_id)`

Deletes the specified Task.

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	<i>task_id</i>	The object ID to operate on
----	----------------	-----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the ID given to it is invalid
<code>OS_ERROR</code>	if the OS delete call fails (return value only verified in coverage test)

10.80.2.4 OS_TaskExit() `void OS_TaskExit (`
 `void)`

Exits the calling task.

The calling thread is terminated. This function does not return.

10.80.2.5 OS_TaskFindIdBySystemData() `int32 OS_TaskFindIdBySystemData (`
 `osal_id_t * task_id,`
 `const void * sysdata,`
 `size_t sysdata_size)`

Reverse-lookup the OSAL task ID from an operating system ID.

This provides a method by which an external entity may find the OSAL task ID corresponding to a system-defined identifier (e.g. TASK_ID, pthread_t, rtems_id, etc).

Normally OSAL does not expose the underlying OS-specific values to the application, but in some circumstances, such as exception handling, the OS may provide this information directly to a BSP handler outside of the normal OSAL API.

Parameters

out	<i>task_id</i>	The buffer where the task id output is stored (must not be null)
in	<i>sysdata</i>	Pointer to the system-provided identification data
in	<i>sysdata_size</i>	Size of the system-provided identification data

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution. (return value only verified in coverage test)
<code>OS_INVALID_POINTER</code>	if a pointer argument is NULL

```
10.80.2.6 OS_TaskGetId() osal_id_t OS_TaskGetId (
    void )
```

Obtain the task id of the calling task.

This function returns the task id of the calling task

Returns

Task ID, or zero if the operation failed (zero is never a valid task ID)

```
10.80.2.7 OS_TaskGetIdByName() int32 OS_TaskGetIdByName (
    osal_id_t * task_id,
    const char * task_name )
```

Find an existing task ID by name.

This function tries to find a task Id given the name of a task

Parameters

out	<i>task_id</i>	will be set to the ID of the existing resource
in	<i>task_name</i>	the name of the existing resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if the pointers passed in are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than OS_MAX_API_NAME
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name wasn't found in the table

```
10.80.2.8 OS_TaskGetInfo() int32 OS_TaskGetInfo (
    osal_id_t task_id,
    OS_task_prop_t * task_prop )
```

Fill a property object buffer with details regarding the resource.

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	<i>task_id</i>	The object ID to operate on
out	<i>task_prop</i>	The property object buffer to fill (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the ID passed to it is invalid
<i>OS_INVALID_POINTER</i>	if the task_prop pointer is NULL

10.80.2.9 OS_TaskInstallDeleteHandler() `int32 OS_TaskInstallDeleteHandler (osal_task_entry function_pointer)`

Installs a handler for when the task is deleted.

This function is used to install a callback that is called when the task is deleted. The callback is called when `OS_TaskDelete` is called with the task ID. A task delete handler is useful for cleaning up resources that a task creates, before the task is removed from the system.

Parameters

<i>in</i>	<i>function_pointer</i>	function to be called when task exits
-----------	-------------------------	---------------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_ERR_INVALID_ID</i>	if the calling context is not an OSAL task
--------------------------	--

10.80.2.10 OS_TaskSetPriority() `int32 OS_TaskSetPriority (osal_id_t task_id, osal_priority_t new_priority)`

Sets the given task to a new priority.

Parameters

<i>in</i>	<i>task_id</i>	The object ID to operate on
<i>in</i>	<i>new_priority</i>	Set the new priority

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the ID passed to it is invalid

Return values

<i>OS_ERR_INVALID_PRIORITY</i>	if the priority is greater than the max allowed (return value only verified in coverage test)
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

10.81 OSAL Time Base APIs

Functions

- `int32 OS_TimeBaseCreate (osal_id_t *timebase_id, const char *timebase_name, OS_TimerSync_t external_sync)`
Create an abstract Time Base resource.
- `int32 OS_TimeBaseSet (osal_id_t timebase_id, uint32 start_time, uint32 interval_time)`
Sets the tick period for simulated time base objects.
- `int32 OS_TimeBaseDelete (osal_id_t timebase_id)`
Deletes a time base object.
- `int32 OS_TimeBaseGetIdByName (osal_id_t *timebase_id, const char *timebase_name)`
Find the ID of an existing time base resource.
- `int32 OS_TimeBaseGetInfo (osal_id_t timebase_id, OS_timebase_prop_t *timebase_prop)`
Obtain information about a timebase resource.
- `int32 OS_TimeBaseGetFreeRun (osal_id_t timebase_id, uint32 *freerun_val)`
Read the value of the timebase free run counter.

10.81.1 Detailed Description

10.81.2 Function Documentation

10.81.2.1 OS_TimeBaseCreate() `int32 OS_TimeBaseCreate (`
 `osal_id_t * timebase_id,`
 `const char * timebase_name,`
 `OS_TimerSync_t external_sync)`

Create an abstract Time Base resource.

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.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

<code>out</code>	<code>timebase_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>timebase_name</code>	The name of the time base (must not be null)
<code>in</code>	<code>external_sync</code>	A synchronization function for BSP hardware-based timer ticks

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_NAME_TAKEN</i>	if the name specified is already used
<i>OS_ERR_NO_FREE_IDS</i>	if there can be no more timebase resources created
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_ERR_NAME_TOO_LONG</i>	if the timebase_name is too long
<i>OS_INVALID_POINTER</i>	if a pointer argument is NULL

10.81.2.2 OS_TimeBaseDelete() `int32 OS_TimeBaseDelete (osal_id_t timebase_id)`

Deletes a time base object.

The helper task and any other resources associated with the time base abstraction will be freed.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timebase_id</i>	The timebase resource to delete
----	--------------------	---------------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

10.81.2.3 OS_TimeBaseGetFreeRun() `int32 OS_TimeBaseGetFreeRun (osal_id_t timebase_id, uint32 * freerun_val)`

Read the value of the timebase free run counter.

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	<i>timebase_id</i>	The timebase to operate on
out	<i>freerun_val</i>	Buffer to store the free run counter (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_INVALID_POINTER</i>	if pointer argument is NULL

```
10.81.2.4 OS_TimeBaseGetIdByName() int32 OS_TimeBaseGetIdByName (
    osal_id_t * timebase_id,
    const char * timebase_name )
```

Find the ID of an existing time base resource.

Given a time base name, find and output the ID associated with it.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

out	<i>timebase_id</i>	will be set to the non-zero ID of the matching resource (must not be null)
in	<i>timebase_name</i>	The name of the timebase resource to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if timebase_id or timebase_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>

Return values

<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.81.2.5 OS_TimeBaseGetInfo() int32 OS_TimeBaseGetInfo (
    osal_id_t timebase_id,
    OS_timebase_prop_t * timebase_prop )
```

Obtain information about a timebase resource.

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.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timebase_id</i>	The timebase resource ID
out	<i>timebase_prop</i>	Buffer to store timebase properties (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_INVALID_POINTER</i>	if the timebase_prop pointer is null
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.81.2.6 OS_TimeBaseSet() int32 OS_TimeBaseSet (
    osal_id_t timebase_id,
    uint32 start_time,
    uint32 interval_time )
```

Sets the tick period for simulated time base objects.

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.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timebase_id</i>	The timebase resource to configure
in	<i>start_time</i>	The amount of delay for the first tick, in microseconds.
in	<i>interval_time</i>	The amount of delay between ticks, in microseconds.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_TIMER_ERR_INVALID_ARGS</i>	if start_time or interval_time are out of range

10.82 OSAL Timer APIs

Functions

- `int32 OS_TimerCreate (osal_id_t *timer_id, const char *timer_name, uint32 *clock_accuracy, OS_TimerCallback_t callback_ptr)`
Create a timer object.
- `int32 OS_TimerAdd (osal_id_t *timer_id, const char *timer_name, osal_id_t timebase_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`
Add a timer object based on an existing TimeBase resource.
- `int32 OS_TimerSet (osal_id_t timer_id, uint32 start_time, uint32 interval_time)`
Configures a periodic or one shot timer.
- `int32 OS_TimerDelete (osal_id_t timer_id)`
Deletes a timer resource.
- `int32 OS_TimerGetIdByName (osal_id_t *timer_id, const char *timer_name)`
Locate an existing timer resource by name.
- `int32 OS_TimerGetInfo (osal_id_t timer_id, OS_timer_prop_t *timer_prop)`
Gets information about an existing timer.

10.82.1 Detailed Description

10.82.2 Function Documentation

10.82.2.1 OS_TimerAdd() `int32 OS_TimerAdd (`
 `osal_id_t * timer_id,`
 `const char * timer_name,`
 `osal_id_t timebase_id,`
 `OS_ArgCallback_t callback_ptr,`
 `void * callback_arg)`

Add a timer object based on an existing TimeBase resource.

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.

The callback function for this method should be declared according to the `OS_ArgCallback_t` function pointer type. The `timer_id` is passed in to the function by the OSAL, and the `arg` parameter is passed through from the `callback_arg` argument on this call.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

See also

[OS_ArgCallback_t](#)

Parameters

<i>out</i>	<i>timer_id</i>	Will be set to the non-zero resource ID of the timer object (must not be null)
<i>in</i>	<i>timer_name</i>	Name of the timer object (must not be null)
<i>in</i>	<i>timebase_id</i>	The time base resource to use as a reference
<i>in</i>	<i>callback_ptr</i>	Application-provided function to invoke (must not be null)
<i>in</i>	<i>callback_arg</i>	Opaque argument to pass to callback function, may be NULL

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if any parameters are NULL
OS_ERR_INVALID_ID	if the timebase_id parameter is not valid
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NAME_TAKEN	if the name is already in use by another timer.
OS_ERR_NO_FREE_IDS	if all of the timers are already allocated.
OS_ERR_INCORRECT_OBJ_STATE	if invoked from a timer context
OS_TIMER_ERR_INTERNAL	if there was an error programming the OS timer (return value only verified in coverage test)

```
10.82.2.2 OS_TimerCreate() int32 OS_TimerCreate (
    osal_id_t * timer_id,
    const char * timer_name,
    uint32 * clock_accuracy,
    OS_TimerCallback_t callback_ptr )
```

Create a timer object.

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. The callback function should be declared according to the `OS_TimerCallback_t` function pointer type. The `timer_id` value is passed to the callback function.

Note

`clock_accuracy` comes from the underlying OS tick value. The nearest integer microsecond value is returned, so may not be exact.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

See also

[OS_TimerCallback_t](#)

Parameters

out	<i>timer_id</i>	Will be set to the non-zero resource ID of the timer object (must not be null)
in	<i>timer_name</i>	Name of the timer object (must not be null)
out	<i>clock_accuracy</i>	Expected precision of the timer, in microseconds. This is the underlying tick value rounded to the nearest microsecond integer. (must not be null)
in	<i>callback_ptr</i>	The function pointer of the timer callback (must not be null).

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if any parameters are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_TAKEN</i>	if the name is already in use by another timer.
<i>OS_ERR_NO_FREE_IDS</i>	if all of the timers are already allocated.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if invoked from a timer context
<i>OS_TIMER_ERR_INTERNAL</i>	if there was an error programming the OS timer (return value only verified in coverage test)

10.82.2.3 OS_TimerDelete() `int32 OS_TimerDelete (osal_id_t timer_id)`

Deletes a timer resource.

The application callback associated with the timer will be stopped, and the resources freed for future use.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timer_id</i>	The timer ID to operate on
----	-----------------	----------------------------

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the timer_id is invalid.
<i>OS_TIMER_ERR_INTERNAL</i>	if there was a problem deleting the timer in the host OS (return value only verified in coverage test)
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.82.2.4 OS_TimerGetIdByName() int32 OS_TimerGetIdByName (
    osal_id_t * timer_id,
    const char * timer_name )
```

Locate an existing timer resource by name.

Outputs the ID associated with the given timer, if it exists.

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

out	<i>timer_id</i>	Will be set to the timer ID corresponding to the name (must not be null)
in	<i>timer_name</i>	The timer name to find (must not be null)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if timer_id or timer_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than OS_MAX_API_NAME
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.82.2.5 OS_TimerGetInfo() int32 OS_TimerGetInfo (
```

```
    osal_id_t timer_id,
    OS_timer_prop_t * timer_prop )
```

Gets information about an existing timer.

This function takes timer_id, and looks it up in the OS table. It puts all of the information known about that timer into a structure pointer to by timer_prop.

Parameters

Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timer_id</i>	The timer ID to operate on
out	<i>timer_prop</i>	<p>Buffer containing timer properties (must not be null)</p> <ul style="list-style-type: none"> • creator: the OS task ID of the task that created this timer • name: the string name of the timer • start_time: the start time in microseconds, if any • interval_time: the interval time in microseconds, if any • accuracy: the accuracy of the timer in microseconds

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timer
OS_INVALID_POINTER	if the timer_prop pointer is null
OS_ERR_INCORRECT_OBJ_STATE	if called from timer/timebase context

10.82.2.6 OS_TimerSet() `int32 OS_TimerSet(`

```
    osal_id_t timer_id,
    uint32 start_time,
    uint32 interval_time )
```

Configures a periodic or one shot timer.

This function programs the timer with a start time and an optional interval time. The start time is the time in microseconds when the user callback function will be called. If the interval time is non-zero, the timer will be reprogrammed with that interval in microseconds to call the user callback function periodically. If the start time and interval time are zero, the function will return an error.

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.

Note

The resolution of the times specified is limited to the clock accuracy returned in the OS_TimerCreate call. If the times specified in the start_msec or interval_msec parameters are less than the accuracy, they will be rounded up to the accuracy of the timer.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

Parameters

in	<i>timer_id</i>	The timer ID to operate on
in	<i>start_time</i>	Time in microseconds to the first expiration
in	<i>interval_time</i>	Time in microseconds between subsequent intervals, value of zero will only call the user callback function once after the start_msec time.

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the timer_id is not valid.
<i>OS_TIMER_ERR_INTERNAL</i>	if there was an error programming the OS timer (return value only verified in coverage test)
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_TIMER_ERR_INVALID_ARGS</i>	if the start_time or interval_time is out of range, or both 0

11 Data Structure Documentation

11.1 CCSDS_ExtendedHeader Struct Reference

CCSDS packet extended header.

```
#include <ccsds_hdr.h>
```

Data Fields

- **uint8 Subsystem [2]**
subsystem qualifier
- **uint8 SystemId [2]**
system qualifier

11.1.1 Detailed Description

CCSDS packet extended header.

Definition at line 73 of file ccsds_hdr.h.

11.1.2 Field Documentation

11.1.2.1 Subsystem `uint8 CCSDS_ExtendedHeader::Subsystem[2]`

subsystem qualifier

Definition at line 75 of file ccsds_hdr.h.

11.1.2.2 SystemId `uint8 CCSDS_ExtendedHeader::SystemId[2]`

system qualifier

Definition at line 82 of file ccsds_hdr.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/msg/fsw/inc/ccsds_hdr.h`

11.2 CCSDS_PrimaryHeader Struct Reference

CCSDS packet primary header.

```
#include <ccsds_hdr.h>
```

Data Fields

- **uint8 StreamId [2]**
packet identifier word (stream ID)
- **uint8 Sequence [2]**
packet sequence word
- **uint8 Length [2]**
packet length word

11.2.1 Detailed Description

CCSDS packet primary header.

Definition at line 51 of file ccsds_hdr.h.

11.2.2 Field Documentation

11.2.2.1 Length `uint8 CCSDS_PrimaryHeader::Length[2]`

packet length word

Definition at line 71 of file `ccsds_hdr.h`.

11.2.2.2 Sequence `uint8 CCSDS_PrimaryHeader::Sequence[2]`

packet sequence word

Definition at line 66 of file `ccsds_hdr.h`.

11.2.2.3 StreamId `uint8 CCSDS_PrimaryHeader::StreamId[2]`

packet identifier word (stream ID)

Definition at line 59 of file `ccsds_hdr.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/msg/fsw/inc/ccsds_hdr.h`

11.3 CF_AbandonCmd Struct Reference

Abandon command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CF_Transaction_Payload_t Payload`

11.3.1 Detailed Description

Abandon command structure.

For command details see [CF_ABANDON_CC](#)

Definition at line 280 of file `default_cf_msgstruct.h`.

11.3.2 Field Documentation

11.3.2.1 CommandHeader `CFE_MSG_CommandHeader_t CF_AbandonCmd::CommandHeader`

Command header.

Definition at line 282 of file `default_cf_msgstruct.h`.

11.3.2.2 Payload `CF_Transaction_Payload_t CF_AbandonCmd::Payload`

Definition at line 283 of file `default_cf_msgstruct.h`.

Referenced by `CF_AbandonCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

11.4 CF_AppData_t Struct Reference

The CF application global state structure.

```
#include <cf_app.h>
```

Data Fields

- `CF_HkPacket_t hk`
- `uint32 RunStatus`
- `CFE_SB_PipeId_t CmdPipe`
- `CFE_TBL_Handle_t config_handle`
- `CF_ConfigTable_t * config_table`
- `CF_Engine_t engine`

11.4.1 Detailed Description

The CF application global state structure.

This contains all variables related to CF application state

Definition at line 80 of file cf_app.h.

11.4.2 Field Documentation

11.4.2.1 CmdPipe `CFE_SB_PipeId_t` CF_AppData_t::CmdPipe

Definition at line 90 of file cf_app.h.

Referenced by CF_AppInit(), and CF_AppMain().

11.4.2.2 config_handle `CFE_TBL_Handle_t` CF_AppData_t::config_handle

Definition at line 92 of file cf_app.h.

Referenced by CF_CheckTables(), and CF_TableInit().

11.4.2.3 config_table `CF_ConfigTable_t*` CF_AppData_t::config_table

Definition at line 93 of file cf_app.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), CF_CFDP_CycleTx(), CF_CFDP_HandleNotKeepFile(), CF_CFDP_InitEngine(), CF_CFDP_IsPollingDir(), CF_CFDP_MsgOutGet(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_Complete(), CF_CFDP_R_Init(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_S_SendFileData(), CF_CFDP_S_Tick(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFileInitiate(), CF_CheckTables(), CF_DoEnableDisableDequeue(), CF_DoEnableDisablePolldir(), CF_GetSetParamCmd(), CF_TableInit(), CF_Timer_Sec2Ticks(), and CF_ValidateMaxOutgoingCmd().

11.4.2.4 engine `CF_Engine_t` CF_AppData_t::engine

Definition at line 95 of file cf_app.h.

Referenced by CF_CFDP_CycleEngine(), CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_InitEngine(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_ResetTransaction(), CF_CFDP_Send(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), CF_CheckTables(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(),

CF_DisableEngineCmd(), CF_DoPurgeQueue(), CF_EnableEngineCmd(), CF_FindTransactionBySequenceNumberAllChannels(), CF_FreeTransaction(), CF_InsertSortPrio(), CF_MoveTransaction(), CF_TraverseAllTransactions_AllChannels(), CF_TsnChanAction(), and CF_WriteQueueCmd().

11.4.2.5 **hk** `CF_HkPacket_t` `CF_AppData_t::hk`

Definition at line 86 of file cf_app.h.

Referenced by CF_AbandonCmd(), CF_AppInit(), CF_AppPipe(), CF_CancelCmd(), CF_CFDP_CycleEngine(), CF_CFDP_DisableEngine(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvDrop(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_Send(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DisableEngineCmd(), CF_DoFreezeThaw(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_MoveTransaction(), CF_NoopCmd(), CF_PlaybackDirCmd(), CF_ProcessGroundCommand(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_SendHkCmd(), CF_ThawCmd(), CF_TxFileCmd(), and CF_WriteQueueCmd().

11.4.2.6 **RunStatus** `uint32` `CF_AppData_t::RunStatus`

Definition at line 88 of file cf_app.h.

Referenced by CF_AppInit(), CF_AppMain(), and CF_CheckTables().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_app.h`

11.5 CF_CancelCmd Struct Reference

Cancel command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CF_Transaction_Payload_t Payload`

11.5.1 Detailed Description

Cancel command structure.

For command details see [CF_CANCEL_CC](#)

Definition at line 269 of file default_cf_msgstruct.h.

11.5.2 Field Documentation

11.5.2.1 **CommandHeader** `CFE_MSG_CommandHeader_t` `CF_CancelCmd::CommandHeader`

Command header.

Definition at line 271 of file default_cf_msgstruct.h.

11.5.2.2 Payload `CF_Transaction_Payload_t` `CF_CancelCmd::Payload`

Definition at line 272 of file default_cf_msgstruct.h.

Referenced by `CF_CancelCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

11.6 CF_CFDP_CycleTx_args Struct ReferenceStructure for use with the `CF_CFDP_CycleTx()` function.

#include <cf_cfdp.h>

Data Fields

- `CF_Channel_t * chan`
channel structure
- `int ran_one`
should be set to 1 if a transaction was cycled

11.6.1 Detailed DescriptionStructure for use with the `CF_CFDP_CycleTx()` function.

Definition at line 34 of file cf_cfdp.h.

11.6.2 Field Documentation**11.6.2.1 chan** `CF_Channel_t*` `CF_CFDP_CycleTx_args::chan`

channel structure

Definition at line 36 of file cf_cfdp.h.

Referenced by `CF_CFDP_CycleTxFirstActive()`.**11.6.2.2 ran_one** `int` `CF_CFDP_CycleTx_args::ran_one`

should be set to 1 if a transaction was cycled

Definition at line 37 of file cf_cfdp.h.

Referenced by `CF_CFDP_CycleTx()`, and `CF_CFDP_CycleTxFirstActive()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp.h`

11.7 CF_CFDP_FileDirectiveDispatchTable_t Struct Reference

A table of receive handler functions based on file directive code.

#include <cf_cfdp_dispatch.h>

Data Fields

- `CF_CFDP_StateRecvFunc_t fdirective` [`CF_CFDP_FileDirective_INVALID_MAX`]
a separate recv handler for each possible file directive PDU in this state

11.7.1 Detailed Description

A table of receive handler functions based on file directive code.

For PDUs identified as a "file directive" type - generally anything other than file data - this provides a table to branch to a different handler function depending on the value of the file directive code.

Definition at line 87 of file cf_cfdp_dispatch.h.

11.7.2 Field Documentation

11.7.2.1 fdirective [CF_CFDP_StateRecvFunc_t](#) CF_CFDP_FileDirectiveDispatchTable_t::fdirective[[CF_CFDP_FileDirective](#)]

a separate recv handler for each possible file directive PDU in this state

Definition at line 90 of file cf_cfdp_dispatch.h.

Referenced by CF_CFDP_R1_Recv(), CF_CFDP_R2_Recv(), CF_CFDP_R_DispatchRecv(), CF_CFDP_S2_Recv(), and CF_CFDP_S_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_cfdp_dispatch.h](#)

11.8 CF_CFDP_Lv Struct Reference

Structure representing CFDP LV Object format.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint8_t length](#)

Length of data field.

11.8.1 Detailed Description

Structure representing CFDP LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

Defined per table 5-2 of CCSDS 727.0-B-5

Definition at line 164 of file cf_cfdp_pdu.h.

11.8.2 Field Documentation

11.8.2.1 length [CF_CFDP_uint8_t](#) CF_CFDP_lv::length

Length of data field.

Definition at line 166 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeLV(), and CF_CFDP_EncodeLV().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_cfdp_pdu.h](#)

11.9 CF_CFDP_PduAck Struct Reference

Structure representing CFDP Acknowledge PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint8_t directive_and_subtype_code](#)
- [CF_CFDP_uint8_t cc_and_transaction_status](#)

11.9.1 Detailed Description

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Definition at line 319 of file cf_cfdp_pdu.h.

11.9.2 Field Documentation**11.9.2.1 cc_and_transaction_status [CF_CFDP_uint8_t](#) CF_CFDP_PduAck::cc_and_transaction_status**

Definition at line 322 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

11.9.2.2 directive_and_subtype_code [CF_CFDP_uint8_t](#) CF_CFDP_PduAck::directive_and_subtype_code

Definition at line 321 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.10 CF_CFDP_PduEof Struct Reference

Structure representing CFDP End of file PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint8_t cc](#)
- [CF_CFDP_uint32_t crc](#)
- [CF_CFDP_uint32_t size](#)

11.10.1 Detailed Description

Structure representing CFDP End of file PDU.

Defined per section 5.2.2 / table 5-6 of CCSDS 727.0-B-5

Definition at line 297 of file cf_cfdp_pdu.h.

11.10.2 Field Documentation**11.10.2.1 cc [CF_CFDP_uint8_t](#) CF_CFDP_PduEof::cc**

Definition at line 299 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_EncodeEof().

11.10.2.2 crc `CF_CFDP_uint32_t` CF_CFDP_PduEof::crc

Definition at line 300 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_EncodeEof().

11.10.2.3 size `CF_CFDP_uint32_t` CF_CFDP_PduEof::size

Definition at line 301 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_EncodeEof().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.11 CF_CFDP_PduFileDataContent Struct Reference

PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `uint8 data [CF_MAX_PDU_SIZE - sizeof(CF_CFDP_PduFileDataHeader_t) - CF_CFDP_MIN_HEADER_SIZE]`

11.11.1 Detailed Description

PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command. This definition allows for the largest data block possible, as CF_MAX_PDU_SIZE - the minimum possible header size. In practice the outgoing file chunk size is limited by whichever is smaller; the remaining data, remaining space in the packet, and outgoing_file_chunk_size.

Definition at line 380 of file cf_cfdp_pdu.h.

11.11.2 Field Documentation

11.11.2.1 data

```
uint8 CF_CFDP_PduFileDataContent::data[CF_MAX_PDU_SIZE - sizeof(CF_CFDP_PduFileDataHeader_t)  
- CF_CFDP_MIN_HEADER_SIZE]
```

Definition at line 382 of file cf_cfdp_pdu.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.12 CF_CFDP_PduFileDataHeader Struct Reference

PDU file data header.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `CF_CFDP_uint32_t offset`

11.12.1 Detailed Description

PDU file data header.

Definition at line 361 of file cf_cfdp_pdu.h.

11.12.2 Field Documentation

11.12.2.1 offset `CF_CFDP_uint32_t` CF_CFDP_PduFileDataHeader::offset

NOTE: while this is the only fixed/required field in the data PDU, it may have segment metadata prior to this, depending on how the fields in the base header are set

Definition at line 368 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeFileDataHeader(), and CF_CFDP_EncodeFileDataHeader().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.13 CF_CFDP_PduFileDirectiveHeader Struct Reference

Structure representing CFDP File Directive Header.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `CF_CFDP_uint8_t directive_code`

11.13.1 Detailed Description

Structure representing CFDP File Directive Header.

Defined per section 5.2 of CCSDS 727.0-B-5

Definition at line 151 of file cf_cfdp_pdu.h.

11.13.2 Field Documentation

11.13.2.1 directive_code `CF_CFDP_uint8_t` CF_CFDP_PduFileDirectiveHeader::directive_code

Definition at line 153 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeFileDirectiveHeader(), and CF_CFDP_EncodeFileDirectiveHeader().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.14 CF_CFDP_PduFin Struct Reference

Structure representing CFDP Finished PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `CF_CFDP_uint8_t flags`

11.14.1 Detailed Description

Structure representing CFDP Finished PDU.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

Definition at line 309 of file cf_cfdp_pdu.h.

11.14.2 Field Documentation

11.14.2.1 flags CF_CFDP_uint8_t CF_CFDP_PduFin::flags

Definition at line 311 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeFin(), and CF_CFDP_EncodeFin().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_pdu.h](#)

11.15 CF_CFDP_PduHeader Struct Reference

Structure representing base CFDP PDU header.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint8_t flags](#)
Flags indicating the PDU type, direction, mode, etc.
- [CF_CFDP_uint16_t length](#)
Length of the entire PDU, in octets.
- [CF_CFDP_uint8_t eid_tsn_lengths](#)
Lengths of the EID+TSN data (bitfields)

11.15.1 Detailed Description

Structure representing base CFDP PDU header.

This header appears at the beginning of all CFDP PDUs, of all types. Note that the header is variable length, it also contains source and destination entity IDs, and the transaction sequence number.

Defined per section 5.1 of CCSDS 727.0-B-5

Note

this contains variable length data for the EID+TSN, which is *not* included in this definition. As a result, the sizeof(\leftarrow CF_CFDP_PduHeader_t) reflects only the size of the fixed fields. Use CF_HeaderSize() to get the actual size of this structure.

Definition at line 137 of file cf_cfdp_pdu.h.

11.15.2 Field Documentation

11.15.2.1 eid_tsn_lengths CF_CFDP_uint8_t CF_CFDP_PduHeader::eid_tsn_lengths

Lengths of the EID+TSN data (bitfields)

Definition at line 141 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

11.15.2.2 flags CF_CFDP_uint8_t CF_CFDP_PduHeader::flags

Flags indicating the PDU type, direction, mode, etc.

Definition at line 139 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

11.15.2.3 length `CF_CFDP_uint16_t` `CF_CFDP_PduHeader::length`

Length of the entire PDU, in octets.

Definition at line 140 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderFinalSize().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.16 CF_CFDP_PduMd Struct Reference

Structure representing CFDP Metadata PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `CF_CFDP_uint8_t segmentation_control`
- `CF_CFDP_uint32_t size`

11.16.1 Detailed Description

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

Definition at line 352 of file cf_cfdp_pdu.h.

11.16.2 Field Documentation

11.16.2.1 segmentation_control

`CF_CFDP_uint8_t CF_CFDP_PduMd::segmentation_control`

Definition at line 354 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

11.16.2.2 size

`CF_CFDP_uint32_t CF_CFDP_PduMd::size`

Definition at line 355 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.17 CF_CFDP_PduNak Struct Reference

Structure representing CFDP Non-Acknowledge PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `CF_CFDP_uint32_t scope_start`
- `CF_CFDP_uint32_t scope_end`

11.17.1 Detailed Description

Structure representing CFDP Non-Acknowledge PDU.

Defined per section 5.2.6 / table 5-10 of CCSDS 727.0-B-5

Definition at line 341 of file cf_cfdp_pdu.h.

11.17.2 Field Documentation

11.17.2.1 **scope_end** `CF_CFDP_uint32_t CF_CFDP_PduNak::scope_end`

Definition at line 344 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeNak(), and CF_CFDP_EncodeNak().

11.17.2.2 **scope_start** `CF_CFDP_uint32_t CF_CFDP_PduNak::scope_start`

Definition at line 343 of file cf_cfdp_pdu.h.

Referenced by CF_CFDP_DecodeNak(), and CF_CFDP_EncodeNak().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.18 CF_CFDP_R_SubstateDispatchTable_t Struct Reference

A dispatch table for receive file transactions, receive side.

```
#include <cf_cfdp_dispatch.h>
```

Data Fields

- const `CF_CFDP_FileDirectiveDispatchTable_t * state [CF_RxSubState_NUM_STATES]`

11.18.1 Detailed Description

A dispatch table for receive file transactions, receive side.

This is used for "receive file" transactions upon receipt of a directive PDU. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 99 of file cf_cfdp_dispatch.h.

11.18.2 Field Documentation

11.18.2.1 **state** const `CF_CFDP_FileDirectiveDispatchTable_t* CF_CFDP_R_SubstateDispatchTable_t::state[CF_RxSubState_NUM_STATES]`

Definition at line 101 of file cf_cfdp_dispatch.h.

Referenced by CF_CFDP_R1_Recv(), CF_CFDP_R2_Recv(), and CF_CFDP_R_DispatchRecv().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_dispatch.h`

11.19 CF_CFDP_S_SubstateRecvDispatchTable_t Struct Reference

A dispatch table for send file transactions, receive side.

```
#include <cf_cfdp_dispatch.h>
```

Data Fields

- const `CF_CFDP_FileDirectiveDispatchTable_t * substate [CF_TxSubState_NUM_STATES]`

11.19.1 Detailed Description

A dispatch table for send file transactions, receive side.

This is used for "send file" transactions upon receipt of a directive PDU. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 110 of file cf_cfdp_dispatch.h.

11.19.2 Field Documentation

11.19.2.1 substate const CF_CFDP_FileDirectiveDispatchTable_t* CF_CFDP_S_SubstateRecvDispatchTable_t::substate[CF_TxSubState_NUM_STATES]

Definition at line 112 of file cf_cfdp_dispatch.h.

Referenced by CF_CFDP_S2_Recv(), and CF_CFDP_S_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_dispatch.h](#)

11.20 CF_CFDP_S_SubstateSendDispatchTable_t Struct Reference

A dispatch table for send file transactions, transmit side.

```
#include <cf_cfdp_dispatch.h>
```

Data Fields

- [CF_CFDP_StateSendFunc_t substate \[CF_TxSubState_NUM_STATES\]](#)

11.20.1 Detailed Description

A dispatch table for send file transactions, transmit side.

This is used for "send file" transactions to generate the next PDU to be sent. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 121 of file cf_cfdp_dispatch.h.

11.20.2 Field Documentation

11.20.2.1 substate CF_CFDP_StateSendFunc_t CF_CFDP_S_SubstateSendDispatchTable_t::substate[CF_TxSubState_NUM_STAT

Definition at line 123 of file cf_cfdp_dispatch.h.

Referenced by CF_CFDP_S1_Tx(), CF_CFDP_S2_Tx(), and CF_CFDP_S_DispatchTransmit().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_dispatch.h](#)

11.21 CF_CFDP_SegmentRequest Struct Reference

Structure representing CFDP Segment Request.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint32_t offset_start](#)
- [CF_CFDP_uint32_t offset_end](#)

11.21.1 Detailed Description

Structure representing CFDP Segment Request.
Defined per section 5.2.6 / table 5-11 of CCSDS 727.0-B-5
Definition at line 330 of file cf_cfdp_pdu.h.

11.21.2 Field Documentation

11.21.2.1 offset_end `CF_CFDP_uint32_t` `CF_CFDP_SegmentRequest::offset_end`

Definition at line 333 of file cf_cfdp_pdu.h.
Referenced by `CF_CFDP_DecodeSegmentRequest()`, and `CF_CFDP_EncodeSegmentRequest()`.

11.21.2.2 offset_start `CF_CFDP_uint32_t` `CF_CFDP_SegmentRequest::offset_start`

Definition at line 332 of file cf_cfdp_pdu.h.
Referenced by `CF_CFDP_DecodeSegmentRequest()`, and `CF_CFDP_EncodeSegmentRequest()`.
The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.22 CF_CFDP_Tick_args Struct Reference

Structure for use with the `CF_CFDP_DoTick()` function.

```
#include <cf_cfdp.h>
```

Data Fields

- `CF_Channel_t * chan`
channel structure
- `void(* fn)(CF_Transaction_t *, int *)`
function pointer
- `bool early_exit`
early exit result
- `int cont`
if 1, then re-traverse the list

11.22.1 Detailed Description

Structure for use with the `CF_CFDP_DoTick()` function.
Definition at line 43 of file cf_cfdp.h.

11.22.2 Field Documentation

11.22.2.1 chan `CF_Channel_t*` `CF_CFDP_Tick_args::chan`

channel structure
Definition at line 45 of file cf_cfdp.h.
Referenced by `CF_CFDP_DoTick()`.

11.22.2.2 cont int CF_CFDP_Tick_args::cont
if 1, then re-traverse the list
Definition at line 48 of file cf_cfdp.h.
Referenced by CF_CFDP_DoTick(), and CF_CFDP_TickTransactions().

11.22.2.3 early_exit bool CF_CFDP_Tick_args::early_exit
early exit result
Definition at line 47 of file cf_cfdp.h.
Referenced by CF_CFDP_DoTick(), and CF_CFDP_TickTransactions().

11.22.2.4 fn void(* CF_CFDP_Tick_args::fn) ([CF_Transaction_t](#) *, int *)
function pointer
Definition at line 46 of file cf_cfdp.h.
Referenced by CF_CFDP_DoTick().
The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_cfdp.h](#)

11.23 CF_CFDP_tlv Struct Reference

Structure representing CFDP TLV Object format.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- [CF_CFDP_uint8_t type](#)
Nature of data field.
- [CF_CFDP_uint8_t length](#)
Length of data field.

11.23.1 Detailed Description

Structure representing CFDP TLV Object format.

These Type + Length + Value pairs used in several CFDP PDU types, typically for file storage requests (section 5.4).

Defined per table 5-3 of CCSDS 727.0-B-5

Definition at line 177 of file cf_cfdp_pdu.h.

11.23.2 Field Documentation

11.23.2.1 length [CF_CFDP_uint8_t](#) CF_CFDP_tlv::length
Length of data field.
Definition at line 180 of file cf_cfdp_pdu.h.
Referenced by CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

11.23.2.2 type [CF_CFDP_uint8_t](#) CF_CFDP_tlv::type
Nature of data field.
Definition at line 179 of file cf_cfdp_pdu.h.
Referenced by CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().
The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.24 CF_CFDP_TxnRecvDispatchTable_t Struct Reference

A table of receive handler functions based on transaction state.

```
#include <cf_cfdp_dispatch.h>
```

Data Fields

- `CF_CFDP_StateRecvFunc_t rx [CF_TxnState_INVALID]`
a separate recv handler for each possible file directive PDU in this state

11.24.1 Detailed Description

A table of receive handler functions based on transaction state.

This reflects the main dispatch table for the receive side of a transaction. Each possible state has a corresponding function pointer in the table to implement the PDU receive action(s) associated with that state.

Definition at line 74 of file `cf_cfdp_dispatch.h`.

11.24.2 Field Documentation

11.24.2.1 rx `CF_CFDP_StateRecvFunc_t` `CF_CFDP_TxnRecvDispatchTable_t::rx[CF_TxnState_INVALID]`

a separate recv handler for each possible file directive PDU in this state

Definition at line 77 of file `cf_cfdp_dispatch.h`.

Referenced by `CF_CFDP_DispatchRecv()`, and `CF_CFDP_RxStateDispatch()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_dispatch.h`

11.25 CF_CFDP_TxnSendDispatchTable_t Struct Reference

A table of transmit handler functions based on transaction state.

```
#include <cf_cfdp_dispatch.h>
```

Data Fields

- `CF_CFDP_StateSendFunc_t tx [CF_TxnState_INVALID]`
Transmit handler function.

11.25.1 Detailed Description

A table of transmit handler functions based on transaction state.

This reflects the main dispatch table for the transmit side of a transaction. Each possible state has a corresponding function pointer in the table to implement the PDU transmit action(s) associated with that state.

Definition at line 62 of file `cf_cfdp_dispatch.h`.

11.25.2 Field Documentation

11.25.2.1 tx `CF_CFDP_StateSendFunc_t` `CF_CFDP_TxnSendDispatchTable_t::tx[CF_TxnState_INVALID]`
Transmit handler function.

Definition at line 64 of file `cf_cfdp_dispatch.h`.

Referenced by `CF_CFDP_DispatchTx()`, and `CF_CFDP_TxStateDispatch()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_dispatch.h`

11.26 CF_CFDP_uint16_t Struct Reference

Encoded 16-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `uint8 octets [2]`

11.26.1 Detailed Description

Encoded 16-bit value in the CFDP PDU.

Definition at line 103 of file `cf_cfdp_pdu.h`.

11.26.2 Field Documentation

11.26.2.1 octets `uint8 CF_CFDP_uint16_t::octets[2]`

Definition at line 105 of file `cf_cfdp_pdu.h`.

Referenced by `CF_Codec_Load_uint16()`, and `CF_Codec_Store_uint16()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.27 CF_CFDP_uint32_t Struct Reference

Encoded 32-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `uint8 octets [4]`

11.27.1 Detailed Description

Encoded 32-bit value in the CFDP PDU.

Definition at line 111 of file `cf_cfdp_pdu.h`.

11.27.2 Field Documentation

11.27.2.1 octets `uint8 CF_CFDP_uint32_t::octets[4]`

Definition at line 113 of file `cf_cfdp_pdu.h`.

Referenced by `CF_Codec_Load_uint32()`, and `CF_Codec_Store_uint32()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.28 CF_CFDP_uint64_t Struct Reference

Encoded 64-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `uint8 octets [8]`

11.28.1 Detailed Description

Encoded 64-bit value in the CFDP PDU.

Definition at line 119 of file cf_cfdp_pdu.h.

11.28.2 Field Documentation

11.28.2.1 `octets uint8 CF_CFDP_uint64_t::octets[8]`

Definition at line 121 of file cf_cfdp_pdu.h.

Referenced by `CF_Codec_Load_uint64()`, and `CF_Codec_Store_uint64()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.29 CF_CFDP_uint8_t Struct Reference

Encoded 8-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

Data Fields

- `uint8 octets [1]`

11.29.1 Detailed Description

Encoded 8-bit value in the CFDP PDU.

Definition at line 95 of file cf_cfdp_pdu.h.

11.29.2 Field Documentation

11.29.2.1 `octets uint8 CF_CFDP_uint8_t::octets[1]`

Definition at line 97 of file cf_cfdp_pdu.h.

Referenced by `CF_Codec_Load_uint8()`, and `CF_Codec_Store_uint8()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

11.30 CF_ChAction_BoolArg Struct Reference

An object to use with channel-scope actions requiring only a boolean argument.

```
#include <cf_cmd.h>
```

Data Fields

- bool [barg](#)

11.30.1 Detailed Description

An object to use with channel-scope actions requiring only a boolean argument.

Definition at line 58 of file cf_cmd.h.

11.30.2 Field Documentation**11.30.2.1 barg bool CF_ChanAction_BoolArg::barg**

Definition at line 60 of file cf_cmd.h.

Referenced by CF_DoEnableDisableDequeue(), and CF_DoFreezeThaw().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cmd.h](#)

11.31 CF_ChanAction_BoolMsgArg Struct Reference

An object to use with channel-scope actions that require the message value.

```
#include <cf_cmd.h>
```

Data Fields

- const [CF_UnionArgs_Payload_t](#) * [data](#)
- bool [barg](#)

11.31.1 Detailed Description

An object to use with channel-scope actions that require the message value.

This combines a boolean action arg with the command message value

Definition at line 87 of file cf_cmd.h.

11.31.2 Field Documentation**11.31.2.1 barg bool CF_ChanAction_BoolMsgArg::barg**

Definition at line 90 of file cf_cmd.h.

Referenced by CF_DoEnableDisablePolldir().

11.31.2.2 data const CF_UnionArgs_Payload_t* CF_ChanAction_BoolMsgArg::data

Definition at line 89 of file cf_cmd.h.

Referenced by CF_DoEnableDisablePolldir().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cmd.h](#)

11.32 CF_ChanAction_MsgArg Struct Reference

An object to use with channel-scope actions that require the message value.

```
#include <cf_cmd.h>
```

Data Fields

- const [CF_UnionArgs_Payload_t](#) * **data**

11.32.1 Detailed Description

An object to use with channel-scope actions that require the message value.
This combines a boolean action arg with the command message value
Definition at line 98 of file cf_cmd.h.

11.32.2 Field Documentation**11.32.2.1 **data** const [CF_UnionArgs_Payload_t](#)* CF_ChAction_MsgArg::data**

Definition at line 100 of file cf_cmd.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cmd.h](#)

11.33 CF_ChAction_SuspResArg Struct Reference

An object to use with channel-scope actions for suspend/resume.

```
#include <cf_cmd.h>
```

Data Fields

- int [same](#)
- bool [action](#)

11.33.1 Detailed Description

An object to use with channel-scope actions for suspend/resume.
This combines a boolean action arg with an output that indicates if it was a change or not.
Definition at line 76 of file cf_cmd.h.

11.33.2 Field Documentation**11.33.2.1 **action** bool CF_ChAction_SuspResArg::action**

Definition at line 79 of file cf_cmd.h.

Referenced by [CF_DoSuspRes_Txn\(\)](#).

11.33.2.2 **same int CF_ChAction_SuspResArg::same**

out param – indicates at least one action was set to its current value

Definition at line 78 of file cf_cmd.h.

Referenced by [CF_DoSuspRes\(\)](#), and [CF_DoSuspRes_Txn\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cmd.h](#)

11.34 CF_Channel Struct Reference

Channel state object.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `CF_CListNode_t * qs [CF_QueueIdx_NUM]`
- `CF_CListNode_t * cs [CF_Direction_NUM]`
- `CFE_SB_PipeId_t pipe`
- `uint32 num_cmd_tx`
- `CF_Playback_t playback [CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN]`
- `CF_Poll_t poll [CF_MAX_POLLING_DIR_PER_CHAN]`
- `osal_id_t sem_id`
semaphore id for output pipe
- `const CF_Transaction_t * cur`
current transaction during channel cycle
- `uint8 tick_type`

11.34.1 Detailed Description

Channel state object.

This keeps the state of CF channels

Each CF channel has a separate transaction list, PDU throttle, playback, and poll state, as well as separate addresses on the underlying message transport (e.g. SB).

Definition at line 392 of file cf_cfdp_types.h.

11.34.2 Field Documentation

11.34.2.1 `cs CF_CListNode_t* CF_Channel::cs[CF_Direction_NUM]`

Definition at line 395 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_FindUnusedChunks()`, `CF_CFDP_InitEngine()`, and `CF_CFDP_ResetTransaction()`.

11.34.2.2 `cur const CF_Transaction_t* CF_Channel::cur`

current transaction during channel cycle

Definition at line 408 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_CycleTx()`, `CF_CFDP_CycleTxFirstActive()`, `CF_CFDP_DoTick()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_ReceiveMessage()`, and `CF_CFDP_ResetTransaction()`.

11.34.2.3 `num_cmd_tx uint32 CF_Channel::num_cmd_tx`

Definition at line 399 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_ResetTransaction()`, and `CF_CFDP_TxFile()`.

11.34.2.4 `pipe CFE_SB_PipeId_t CF_Channel::pipe`

Definition at line 397 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_DisableEngine()`, `CF_CFDP_InitEngine()`, and `CF_CFDP_ReceiveMessage()`.

11.34.2.5 playback `CF_Playback_t` CF_Channel::playback[`CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN`]

Definition at line 401 of file cf_cfdp_types.h.

Referenced by CF_CFDP_DisableEngine(), CF_CFDP_PlaybackDir(), and CF_CFDP_ProcessPlaybackDirectories().

11.34.2.6 poll `CF_Poll_t` CF_Channel::poll[`CF_MAX_POLLING_DIR_PER_CHAN`]

Definition at line 404 of file cf_cfdp_types.h.

Referenced by CF_CFDP_DisableEngine(), and CF_CFDP_ProcessPollingDirectories().

11.34.2.7 qs `CF_CListNode_t*` CF_Channel::qs[`CF_QueueIdx_NUM`]

Definition at line 394 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_TickTransactions(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), CF_DoPurgeQueue(), CF_FindTransactionBySequenceNumber(), CF_FindUnusedTransaction(), CF_InsertSortPrio(), CF_MoveTransaction(), CF_TraverseAllTransactions(), CF_WriteHistoryQueueDataToFile(), and CF_WriteTxnQueueDataToFile().

11.34.2.8 sem_id `osal_id_t` CF_Channel::sem_id

semaphore id for output pipe

Definition at line 406 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine(), and CF_CFDP_MsgOutGet().

11.34.2.9 tick_type `uint8` CF_Channel::tick_type

Definition at line 410 of file cf_cfdp_types.h.

Referenced by CF_CFDP_TickTransactions().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_types.h`

11.35 CF_ChannelConfig Struct Reference

Configuration entry for CFDP channel.

```
#include <default_cf_tbldefs.h>
```

Data Fields

- `uint32 max_outgoing_messages_per_wakeup`
max number of messages to send per wakeup (0 - unlimited)
- `uint32 rx_max_messages_per_wakeup`
max number of rx messages to process per wakeup
- `uint32 ack_timer_s`
Acknowledge timer in seconds.
- `uint32 nak_timer_s`
Non-acknowledge timer in seconds.
- `uint32 inactivity_timer_s`
Inactivity timer in seconds.
- `uint8 ack_limit`
- `uint8 nak_limit`
- `CFE_SB_MsgId_Atom_t mid_input`

- **msgid** integer value for incoming messages
- **CFE_SB_MsgId_Atom_t mid_output**
 - msgid integer value for outgoing messages
- **uint16 pipe_depth_input**
 - depth of pipe to receive incoming PDU
- **CF_PollDir_t polldir [CF_MAX_POLLING_DIR_PER_CHAN]**
 - Configuration for polled directories.
- **char sem_name [OS_MAX_API_NAME]**
 - name of throttling semaphore in TO
- **uint8 dequeue_enabled**
 - if 1, then the channel will make pending transactions active
- **char move_dir [OS_MAX_PATH_LEN]**
 - Move directory if not empty.

11.35.1 Detailed Description

Configuration entry for CFDP channel.

Definition at line 57 of file default_cf_tbldefs.h.

11.35.2 Field Documentation

11.35.2.1 ack_limit uint8 CF_ChannelConfig::ack_limit

number of times to retry ACK (for ex, send FIN and wait for fin-ack)

Definition at line 66 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_R_Tick(), CF_CFDP_S_Tick(), and CF_GetSetParamCmd().

11.35.2.2 ack_timer_s uint32 CF_ChannelConfig::ack_timer_s

Acknowledge timer in seconds.

Definition at line 62 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_ArmAckTimer(), and CF_GetSetParamCmd().

11.35.2.3 dequeue_enabled uint8 CF_ChannelConfig::dequeue_enabled

if 1, then the channel will make pending transactions active

Definition at line 77 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_CycleTx(), and CF_DoEnableDisableDequeue().

11.35.2.4 inactivity_timer_s uint32 CF_ChannelConfig::inactivity_timer_s

Inactivity timer in seconds.

Definition at line 64 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_ArmInactTimer(), and CF_GetSetParamCmd().

11.35.2.5 max_outgoing_messages_per_wakeup uint32 CF_ChannelConfig::max_outgoing_messages_per_wakeup

max number of messages to send per wakeup (0 - unlimited)

Definition at line 59 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_MsgOutGet(), and CF_GetSetParamCmd().

11.35.2.6 mid_input `CFE_SB_MsgId_Atom_t` CF_ChannelConfig::mid_input
msgid integer value for incoming messages
Definition at line 69 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_InitEngine().

11.35.2.7 mid_output `CFE_SB_MsgId_Atom_t` CF_ChannelConfig::mid_output
msgid integer value for outgoing messages
Definition at line 70 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_MsgOutGet().

11.35.2.8 move_dir `char` CF_ChannelConfig::move_dir[`OS_MAX_PATH_LEN`]
Move directory if not empty.
Definition at line 78 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_HandleNotKeepFile().

11.35.2.9 nak_limit `uint8` CF_ChannelConfig::nak_limit
number of times to retry NAK before giving up (resets on a single response)
Definition at line 67 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_R2_Complete(), and CF_GetSetParamCmd().

11.35.2.10 nak_timer_s `uint32` CF_ChannelConfig::nak_timer_s
Non-acknowledge timer in seconds.
Definition at line 63 of file default_cf_tbldefs.h.
Referenced by CF_GetSetParamCmd().

11.35.2.11 pipe_depth_input `uint16` CF_ChannelConfig::pipe_depth_input
depth of pipe to receive incoming PDU
Definition at line 72 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_InitEngine().

11.35.2.12 polldir `CF_PollDir_t` CF_ChannelConfig::polldir[`CF_MAX_POLLING_DIR_PER_CHAN`]
Configuration for polled directories.
Definition at line 74 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_IsPollingDir(), CF_CFDP_ProcessPollingDirectories(), and CF_DoEnableDisablePolldir().

11.35.2.13 rx_max_messages_per_wakeup `uint32` CF_ChannelConfig::rx_max_messages_per_wakeup
max number of rx messages to process per wakeup
Definition at line 60 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_ReceiveMessage().

11.35.2.14 sem_name `char` CF_ChannelConfig::sem_name[`OS_MAX_API_NAME`]
name of throttling semaphore in TO
Definition at line 76 of file default_cf_tbldefs.h.
Referenced by CF_CFDP_InitEngine(), and CF_ValidateMaxOutgoingCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_tbldefs.h](#)

11.36 CF_Chunk Struct Reference

Pairs an offset with a size to identify a specific piece of a file.

```
#include <cf_chunk.h>
```

Data Fields

- **CF_Offset_t offset**
The start offset of the chunk within the file.
- **CF_Size_t size**
The size of the chunk.

11.36.1 Detailed Description

Pairs an offset with a size to identify a specific piece of a file.

Definition at line 38 of file cf_chunk.h.

11.36.2 Field Documentation

11.36.2.1 offset [CF_Offset_t](#) CF_Chunk::offset

The start offset of the chunk within the file.

Definition at line 40 of file cf_chunk.h.

Referenced by CF_CFDP_R2_GapCompute(), CF_CFDP_S_CheckAndRespondNak(), CF_ChunkList_ComputeGaps(), CF_ChunkList_RemoveFromFirst(), CF_Chunks_CombineNext(), CF_Chunks_CombinePrevious(), and CF_Chunks_FindInsertPosition().

11.36.2.2 size [CF_Size_t](#) CF_Chunk::size

The size of the chunk.

Definition at line 41 of file cf_chunk.h.

Referenced by CF_CFDP_R2_GapCompute(), CF_CFDP_S_CheckAndRespondNak(), CF_ChunkList_ComputeGaps(), CF_ChunkList_RemoveFromFirst(), CF_Chunks_CombineNext(), CF_Chunks_CombinePrevious(), CF_Chunks_FindSmallestSize(), and CF_Chunks_Insert().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_chunk.h](#)

11.37 CF_ChunkList Struct Reference

A list of CF_Chunk_t pairs.

```
#include <cf_chunk.h>
```

Data Fields

- **CF_Idx_t count**
number of chunks currently in the array
- **CF_Idx_t max_chunks**
maximum number of chunks allowed in the list (allocation size)

- `CF_Chunk_t * chunks`
chunk list array

11.37.1 Detailed Description

A list of CF_Chunk_t pairs.

This list is ordered by chunk offset, from lowest to highest

Definition at line 49 of file cf_chunk.h.

11.37.2 Field Documentation

11.37.2.1 `chunks` `CF_Chunk_t* CF_ChunkList::chunks`

chunk list array

Definition at line 53 of file cf_chunk.h.

Referenced by `CF_ChunkList_ComputeGaps()`, `CF_ChunkList_GetFirstChunk()`, `CF_ChunkList_RemoveFromFirst()`, `CF_ChunkListInit()`, `CF_ChunkListReset()`, `CF_Chunks_CombineNext()`, `CF_Chunks_CombinePrevious()`, `CF_Chunks_EraseChunk()`, `CF_Chunks_EraseRange()`, `CF_Chunks_FindInsertPosition()`, `CF_Chunks_FindSmallestSize()`, `CF_Chunks_Insert()`, and `CF_Chunks_InsertChunk()`.

11.37.2.2 `count` `CF_ChunkIdx_t CF_ChunkList::count`

number of chunks currently in the array

Definition at line 51 of file cf_chunk.h.

Referenced by `CF_CFDP_R_SubstateSendNak()`, `CF_ChunkList_ComputeGaps()`, `CF_ChunkList_GetFirstChunk()`, `CF_ChunkListReset()`, `CF_Chunks_CombineNext()`, `CF_Chunks_EraseChunk()`, `CF_Chunks_EraseRange()`, `CF_Chunks_FindInsertPosition()`, `CF_Chunks_FindSmallestSize()`, `CF_Chunks_Insert()`, and `CF_Chunks_InsertChunk()`.

11.37.2.3 `max_chunks` `CF_ChunkIdx_t CF_ChunkList::max_chunks`

maximum number of chunks allowed in the list (allocation size)

Definition at line 52 of file cf_chunk.h.

Referenced by `CF_CFDP_R_SubstateSendNak()`, `CF_ChunkListInit()`, `CF_ChunkListReset()`, `CF_Chunks_Insert()`, and `CF_Chunks_InsertChunk()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_chunk.h`

11.38 CF_ChunkWrapper Struct Reference

Wrapper around a CF_ChunkList_t object.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `CF_ChunkList_t chunks`
- `CF_CListNode_t cl_node`

11.38.1 Detailed Description

Wrapper around a CF_ChunkList_t object.

This allows a CF_ChunkList_t to be stored within a CList data storage structure

Definition at line 189 of file cf_cfdp_types.h.

11.38.2 Field Documentation

11.38.2.1 chunks `CF_ChunkList_t` CF_ChunkWrapper::chunks

Definition at line 191 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine(), CF_CFDP_R2_Complete(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_S2_Nak(), and CF_CFDP_S_CheckAndRespondNak().

11.38.2.2 cl_node `CF_CListNode_t` CF_ChunkWrapper::cl_node

Definition at line 192 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine(), and CF_CFDP_ResetTransaction().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_cfdp_types.h](#)

11.39 CF_CListNode Struct Reference

Node link structure.

```
#include <cf_clist.h>
```

Data Fields

- struct [CF_CListNode](#) * next
- struct [CF_CListNode](#) * prev

11.39.1 Detailed Description

Node link structure.

Definition at line 52 of file cf_clist.h.

11.39.2 Field Documentation

11.39.2.1 next struct `CF_CListNode*` CF_CListNode::next

Definition at line 54 of file cf_clist.h.

Referenced by CF_CList_InitNode(), CF_CList_InsertAfter(), CF_CList_InsertBack(), CF_CList_InsertFront(), CF_CList_Remove(), and CF_CList_Traverse().

11.39.2.2 prev struct `CF_CListNode*` CF_CListNode::prev

Definition at line 55 of file cf_clist.h.

Referenced by CF_CList_InitNode(), CF_CList_InsertAfter(), CF_CList_InsertBack(), CF_CList_InsertFront(), CF_CList_Remove(), and CF_CList_Traverse_R().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_clist.h](#)

11.40 CF_Codec_BitField Struct Reference

Data Fields

- `uint32 shift`
- `uint32 mask`

11.40.1 Detailed Description

Definition at line 33 of file cf_codec.c.

11.40.2 Field Documentation

11.40.2.1 `mask uint32 CF_Codec_BitField::mask`

Definition at line 36 of file cf_codec.c.

11.40.2.2 `shift uint32 CF_Codec_BitField::shift`

Definition at line 35 of file cf_codec.c.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_codec.c](#)

11.41 CF_CodecState Struct Reference

Tracks the current state of an encode or decode operation.

```
#include <cf_codec.h>
```

Data Fields

- `bool is_valid`
whether decode is valid or not. Set false on end of decode or error condition.
- `size_t next_offset`
Offset of next byte to encode/decode, current position in PDU.
- `size_t max_size`
Maximum number of bytes in the PDU.

11.41.1 Detailed Description

Tracks the current state of an encode or decode operation.

This encapsulates the common state between encode and decode

Definition at line 38 of file cf_codec.h.

11.41.2 Field Documentation

11.41.2.1 `is_valid bool CF_CodecState::is_valid`

whether decode is valid or not. Set false on end of decode or error condition.

Definition at line 40 of file cf_codec.h.

Referenced by CF_CFDP_CodecIsOK(), CF_CFDP_CodecReset(), and CF_CFDP_CodecSetDone().

11.41.2.2 `max_size size_t CF_CodecState::max_size`

Maximum number of bytes in the PDU.

Definition at line 42 of file cf_codec.h.

Referenced by CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetRemain(), CF_CFDP_CodecGetSize(), CF_CFDP_CodecReset(), CF_CFDP_DecodeStart(), and CF_CFDP_EncodeStart().

11.41.2.3 next_offset size_t CF_CodecState::next_offset

Offset of next byte to encode/decode, current position in PDU.

Definition at line 41 of file cf_codec.h.

Referenced by CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetPosition(), CF_CFDP_CodecGetRemain(), and CF_CFDP_CodecReset().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_codec.h](#)

11.42 CF_ConfigTable Struct Reference

Top-level CFDP configuration structure.

```
#include <default_cf_tblstruct.h>
```

Data Fields

- [uint32 ticks_per_second](#)
expected ticks per second to CFDP app
- [uint32 rx_crc_calc_bytes_per_wakeup](#)
max number of bytes per wakeup to calculate r2 CRC for recv'd file (must be 1024-byte aligned)
- [CF_EntityId_t local_eid](#)
the local entity ID of the CF app
- [CF_ChannelConfig_t chan \[CF_NUM_CHANNELS\]](#)
Channel configuration.
- [uint16 outgoing_file_chunk_size](#)
maximum size of outgoing file data chunk in a PDU. Limited by CF_MAX_PDU_SIZE minus the PDU header(s)
- [char tmp_dir \[CF_FILENAME_MAX_PATH\]](#)
directory to put temp files
- [char fail_dir \[CF_FILENAME_MAX_PATH\]](#)
fail directory

11.42.1 Detailed Description

Top-level CFDP configuration structure.

Definition at line 40 of file default_cf_tblstruct.h.

11.42.2 Field Documentation

11.42.2.1 chan [CF_ChannelConfig_t](#) CF_ConfigTable:::chan[CF_NUM_CHANNELS]

Channel configuration.

Definition at line 49 of file default_cf_tblstruct.h.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), CF_CFDP_CycleTx(), CF_CFDP_HandleNotKeepFile(), CF_CFDP_InitEngine(), CF_CFDP_IsPollingDir(), CF_CFDP_MsgOutGet(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_Complete(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_S_Tick(), CF_DoEnableDisableDequeue(), CF_DoEnableDisablePolldir(), CF_GetSetParamCmd(), and CF_ValidateMaxOutgoingCmd().

11.42.2.2 fail_dir `char CF_ConfigTable::fail_dir[CF_FILENAME_MAX_PATH]`
fail directory

Definition at line 54 of file default_cf_tblstruct.h.
Referenced by CF_CFDP_HandleNotKeepFile().

11.42.2.3 local_eid `CF_EntityId_t CF_ConfigTable::local_eid`

the local entity ID of the CF app

Definition at line 47 of file default_cf_tblstruct.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_ReceiveMessage(), CF_CFDP_S_SendFileData(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFile_Initiate(), and CF_GetSetParamCmd().

11.42.2.4 outgoing_file_chunk_size `uint16 CF_ConfigTable::outgoing_file_chunk_size`

maximum size of outgoing file data chunk in a PDU. Limited by CF_MAX_PDU_SIZE minus the PDU header(s)

Definition at line 51 of file default_cf_tblstruct.h.

Referenced by CF_CFDP_S_SendFileData(), CF_GetSetParamCmd(), and CF_ValidateConfigTable().

11.42.2.5 rx_crc_calc_bytes_per_wakeup `uint32 CF_ConfigTable::rx_crc_calc_bytes_per_wakeup`

max number of bytes per wakeup to calculate r2 CRC for recv'd file (must be 1024-byte aligned)

Definition at line 43 of file default_cf_tblstruct.h.

Referenced by CF_GetSetParamCmd(), and CF_ValidateConfigTable().

11.42.2.6 ticks_per_second `uint32 CF_ConfigTable::ticks_per_second`

expected ticks per second to CFDP app

Definition at line 42 of file default_cf_tblstruct.h.

Referenced by CF_GetSetParamCmd(), CF_Timer_Sec2Ticks(), and CF_ValidateConfigTable().

11.42.2.7 tmp_dir `char CF_ConfigTable::tmp_dir[CF_FILENAME_MAX_PATH]`

directory to put temp files

Definition at line 53 of file default_cf_tblstruct.h.

Referenced by CF_CFDP_R_Init().

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_tblstruct.h

11.43 CF_Crc Struct Reference

CRC state object.

```
#include <cf_crc.h>
```

Data Fields

- `uint32 working`
- `uint32 result`
- `uint8 index`

11.43.1 Detailed Description

CRC state object.

Definition at line 34 of file cf_crc.h.

11.43.2 Field Documentation

11.43.2.1 index `uint8 CF_Crc::index`

Definition at line 38 of file cf_crc.h.

Referenced by CF_CRC_Digest(), and CF_CRC_Finalize().

11.43.2.2 result `uint32 CF_Crc::result`

Definition at line 37 of file cf_crc.h.

Referenced by CF_CFDP_R_CheckCrc(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), CF_CRC_Digest(), and CF_CRC_Finalize().

11.43.2.3 working `uint32 CF_Crc::working`

Definition at line 36 of file cf_crc.h.

Referenced by CF_CRC_Digest(), and CF_CRC_Finalize().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_crc.h`

11.44 CF_DecoderState Struct Reference

Current state of a decode operation.

```
#include <cf_codec.h>
```

Data Fields

- `CF_CodecState_t codec_state`
Common state.
- `const uint8 * base`
Pointer to start of encoded PDU data.

11.44.1 Detailed Description

Current state of a decode operation.

State structure for decodes

Definition at line 61 of file cf_codec.h.

11.44.2 Field Documentation

11.44.2.1 base `const uint8* CF_DecoderState::base`

Pointer to start of encoded PDU data.

Definition at line 64 of file cf_codec.h.

Referenced by CF_CFDP_DecodeStart(), and CF_CFDP_DoDecodeChunk().

11.44.2.2 codec_state `CF_CodecState_t` `CF_DecoderState::codec_state`
Common state.

Definition at line 63 of file cf_codec.h.

Referenced by CF_CFDP_DecodeStart(), and CF_CFDP_DoDecodeChunk().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_codec.h](#)

11.45 CF_DisableDequeueCmd Struct Reference

DisableDequeue command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CF_UnionArgs_Payload_t Payload`
Generic command arguments.

11.45.1 Detailed Description

DisableDequeue command structure.

For command details see [CF_DISABLE_DEQUEUE_CC](#)

Definition at line 148 of file default_cf_msgstruct.h.

11.45.2 Field Documentation

11.45.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CF_DisableDequeueCmd::CommandHeader`
Command header.

Definition at line 150 of file default_cf_msgstruct.h.

11.45.2.2 Payload `CF_UnionArgs_Payload_t` `CF_DisableDequeueCmd::Payload`
Generic command arguments.

Definition at line 151 of file default_cf_msgstruct.h.

Referenced by CF_DisableDequeueCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.46 CF_DisableDirPollingCmd Struct Reference

DisableDirPolling command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CF_UnionArgs_Payload_t Payload`
Generic command arguments.

11.46.1 Detailed Description

DisableDirPolling command structure.

For command details see [CF_DISABLE_DIR_POLLING_CC](#)

Definition at line 170 of file default_cf_msgstruct.h.

11.46.2 Field Documentation

11.46.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_DisableDirPollingCmd::CommandHeader
Command header.

Definition at line 172 of file default_cf_msgstruct.h.

11.46.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_DisableDirPollingCmd::Payload
Generic command arguments.

Definition at line 173 of file default_cf_msgstruct.h.

Referenced by CF_DisableDirPollingCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.47 CF_DisableEngineCmd Struct Reference

DisableEngine command structure.

#include <default_cf_msgstruct.h>

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.47.1 Detailed Description

DisableEngine command structure.

For command details see [CF_DISABLE_ENGINE_CC](#)

Definition at line 94 of file default_cf_msgstruct.h.

11.47.2 Field Documentation

11.47.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_DisableEngineCmd::CommandHeader
Command header.

Definition at line 96 of file default_cf_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.48 CF_EnableDequeueCmd Struct Reference

EnableDequeue command structure.

#include <default_cf_msgstruct.h>

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_UnionArgs_Payload_t](#) Payload
Generic command arguments.

11.48.1 Detailed Description

EnableDequeue command structure.

For command details see [CF_ENABLE_DEQUEUE_CC](#)

Definition at line 137 of file default_cf_msgstruct.h.

11.48.2 Field Documentation

11.48.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_EnableDequeueCmd::CommandHeader

Command header.

Definition at line 139 of file default_cf_msgstruct.h.

11.48.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_EnableDequeueCmd::Payload

Generic command arguments.

Definition at line 140 of file default_cf_msgstruct.h.

Referenced by CF_EnableDequeueCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgstruct.h

11.49 CF_EnableDirPollingCmd Struct Reference

EnableDirPolling command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_UnionArgs_Payload_t](#) Payload
Generic command arguments.

11.49.1 Detailed Description

EnableDirPolling command structure.

For command details see [CF_ENABLE_DIR_POLLING_CC](#)

Definition at line 159 of file default_cf_msgstruct.h.

11.49.2 Field Documentation

11.49.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_EnableDirPollingCmd::CommandHeader
Command header.
Definition at line 161 of file default_cf_msgstruct.h.

11.49.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_EnableDirPollingCmd::Payload
Generic command arguments.
Definition at line 162 of file default_cf_msgstruct.h.
Referenced by CF_EnableDirPollingCmd().
The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.50 CF_EnableEngineCmd Struct Reference

EnableEngine command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.

11.50.1 Detailed Description

EnableEngine command structure.

For command details see [CF_ENABLE_ENGINE_CC](#)

Definition at line 84 of file default_cf_msgstruct.h.

11.50.2 Field Documentation

11.50.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_EnableEngineCmd::CommandHeader
Command header.
Definition at line 86 of file default_cf_msgstruct.h.
The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.51 CF_EncoderState Struct Reference

Current state of an encode operation.

```
#include <cf_codec.h>
```

Data Fields

- [CF_CodecState_t codec_state](#)
Common state.
- [uint8 * base](#)
Pointer to start of encoded PDU data.

11.51.1 Detailed Description

Current state of an encode operation.

State structure for encodes

Definition at line 50 of file cf_codec.h.

11.51.2 Field Documentation

11.51.2.1 `base` `uint8* CF_EncoderState::base`

Pointer to start of encoded PDU data.

Definition at line 53 of file cf_codec.h.

Referenced by CF_CFDP_DoEncodeChunk(), CF_CFDP_EncodeHeaderFinalSize(), and CF_CFDP_EncodeStart().

11.51.2.2 `codec_state` `CF_CodecState_t CF_EncoderState::codec_state`

Common state.

Definition at line 52 of file cf_codec.h.

Referenced by CF_CFDP_DoEncodeChunk(), and CF_CFDP_EncodeStart().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_codec.h`

11.52 CF_Engine Struct Reference

An engine represents a pairing to a local EID.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `CF_TransactionSeq_t seq_num`
- `CF_Output_t out`
- `CF_Input_t in`
- `CF_Transaction_t transactions [CF_NUM_TRANSACTIONS]`
- `CF_History_t histories [CF_NUM_HISTORIES]`
- `CF_Channel_t channels [CF_NUM_CHANNELS]`
- `CF_ChunkWrapper_t chunks [CF_NUM_TRANSACTIONS *CF_Direction_NUM]`
- `CF_Chunk_t chunk_mem [CF_NUM_CHUNKS_ALL_CHANNELS]`
- `uint32 outgoing_counter`
- `bool enabled`

11.52.1 Detailed Description

An engine represents a pairing to a local EID.

Each engine can have at most CF_MAX_SIMULTANEOUS_TRANSACTIONS

Definition at line 442 of file cf_cfdp_types.h.

11.52.2 Field Documentation

11.52.2.1 channels `CF_Channel_t` `CF_Engine::channels[CF_NUM_CHANNELS]`

Definition at line 452 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleEngine(), CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_InitEngine(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_Process← PlaybackDirectory(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_ResetTransaction(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), CF_CList_InsertAfter_Ex(), C← F_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), CF_DoPurgeQueue(), CF_Find← TransactionBySequenceNumberAllChannels(), CF_FreeTransaction(), CF_InsertSortPrio(), CF_MoveTransaction(), CF_TraverseAllTransactions_All_Channels(), CF_TsnChanAction(), and CF_WriteQueueCmd().

11.52.2.2 chunk_mem `CF_Chunk_t` `CF_Engine::chunk_mem[CF_NUM_CHUNKS_ALL_CHANNELS]`

Definition at line 455 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine().

11.52.2.3 chunks `CF_ChunkWrapper_t` `CF_Engine::chunks[CF_NUM_TRANSACTIONS *CF_Direction_NUM]`

Definition at line 454 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine().

11.52.2.4 enabled `bool` `CF_Engine::enabled`

Definition at line 458 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleEngine(), CF_CFDP_DisableEngine(), CF_CFDP_InitEngine(), CF_CheckTables(), CF_DisableEngineCmd(), and CF_EnableEngineCmd().

11.52.2.5 histories `CF_History_t` `CF_Engine::histories[CF_NUM_HISTORIES]`

Definition at line 451 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine().

11.52.2.6 in `CF_Input_t` `CF_Engine::in`

Definition at line 447 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ReceiveMessage().

11.52.2.7 out `CF_Output_t` `CF_Engine::out`

Definition at line 446 of file cf_cfdp_types.h.

Referenced by CF_CFDP_MsgOutGet(), and CF_CFDP_Send().

11.52.2.8 outgoing_counter `uint32` `CF_Engine::outgoing_counter`

Definition at line 457 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleEngine(), and CF_CFDP_MsgOutGet().

11.52.2.9 seq_num `CF_TransactionSeq_t` `CF_Engine::seq_num`

Definition at line 444 of file cf_cfdp_types.h.

Referenced by CF_CFDP_TxFile_Initiate().

11.52.2.10 transactions `CF_Transaction_t` CF_Engine::transactions[CF_NUM_TRANSACTIONS]

Definition at line 450 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.53 CF_EotPacket Struct Reference

End of transaction packet.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CF_EotPacket_Payload_t Payload`

11.53.1 Detailed Description

End of transaction packet.

Definition at line 56 of file default_cf_msgstruct.h.

11.53.2 Field Documentation

11.53.2.1 Payload `CF_EotPacket_Payload_t` CF_EotPacket::Payload

Definition at line 59 of file default_cf_msgstruct.h.

Referenced by CF_CFDP_SendEotPkt().

11.53.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` CF_EotPacket::TelemetryHeader

Telemetry header.

Definition at line 58 of file default_cf_msgstruct.h.

Referenced by CF_CFDP_SendEotPkt().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.54 CF_EotPacket_Payload Struct Reference

End of transaction packet.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `CF_TransactionSeq_t seq_num`
transaction identifier, stays constant for entire transfer
- `uint32 channel`
Channel number.
- `uint32 direction`
direction of this transaction
- `uint32 state`

- *Transaction state.*
- `uint32 txn_stat`
final status code of transaction (extended CFDP CC)
- `CF_EntityId_t src_eid`
the source eid of the transaction
- `CF_EntityId_t peer_eid`
peer_eid is always the "other guy", same src_eid for RX
- `uint32 fsize`
File size.
- `uint32 crc_result`
CRC result.
- `CF_TxnFilenames_t fnames`
file names associated with this transaction

11.54.1 Detailed Description

End of transaction packet.

Definition at line 127 of file default_cf_msgdefs.h.

11.54.2 Field Documentation

11.54.2.1 `channel` `uint32` CF_EotPacket_Payload::channel

Channel number.

Definition at line 130 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_SendEotPkt().

11.54.2.2 `crc_result` `uint32` CF_EotPacket_Payload::crc_result

CRC result.

Definition at line 137 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_SendEotPkt().

11.54.2.3 `direction` `uint32` CF_EotPacket_Payload::direction

direction of this transaction

Definition at line 131 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_SendEotPkt().

11.54.2.4 `fnames` `CF_TxnFilenames_t` CF_EotPacket_Payload::fnames

file names associated with this transaction

Definition at line 138 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_SendEotPkt().

11.54.2.5 `fsize` `uint32` CF_EotPacket_Payload::fsize

File size.

Definition at line 136 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_SendEotPkt().

11.54.2.6 peer_eid `CF_EntityId_t` CF_EotPacket_Payload::peer_eid
peer_eid is always the "other guy", same src_eid for RX
Definition at line 135 of file default_cf_msgdefs.h.
Referenced by CF_CFDP_SendEotPkt().

11.54.2.7 seq_num `CF_TransactionSeq_t` CF_EotPacket_Payload::seq_num
transaction identifier, stays constant for entire transfer
Definition at line 129 of file default_cf_msgdefs.h.
Referenced by CF_CFDP_SendEotPkt().

11.54.2.8 src_eid `CF_EntityId_t` CF_EotPacket_Payload::src_eid
the source eid of the transaction
Definition at line 134 of file default_cf_msgdefs.h.
Referenced by CF_CFDP_SendEotPkt().

11.54.2.9 state `uint32` CF_EotPacket_Payload::state
Transaction state.
Definition at line 132 of file default_cf_msgdefs.h.
Referenced by CF_CFDP_SendEotPkt().

11.54.2.10 txn_stat `uint32` CF_EotPacket_Payload::txn_stat
final status code of transaction (extended CFDP CC)
Definition at line 133 of file default_cf_msgdefs.h.
Referenced by CF_CFDP_SendEotPkt().
The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgdefs.h

11.55 CF_Flags_Common Struct Reference

Data that applies to all types of transactions.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `uint8 q_index`
Q index this is in.
- `bool ack_timer_armed`
- `bool suspended`
- `bool canceled`
- `bool crc_calc`

11.55.1 Detailed Description

Data that applies to all types of transactions.
Definition at line 275 of file cf_cfdp_types.h.

11.55.2 Field Documentation

11.55.2.1 ack_timer_armed bool CF_Flags_Common::ack_timer_armed

Definition at line 278 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_R_Tick(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForEofAck(), and CF_CFDP_S_Tick().

11.55.2.2 canceled bool CF_Flags_Common::canceled

Definition at line 280 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CancelTransaction(), and CF_CFDP_R2_Reset().

11.55.2.3 crc_calc bool CF_Flags_Common::crc_calc

Definition at line 281 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_SubstateSendFin(), and CF_CFDP_S_SendEof().

11.55.2.4 q_index uint8 CF_Flags_Common::q_index

Q index this is in.

Definition at line 277 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleTxFirstActive(), CF_CFDP_GetClass(), CF_CFDP_IsSender(), CF_CFDP_ReceiveMessage(), CF_CFDP_ResetTransaction(), CF_DequeueTransaction(), CF_FreeTransaction(), CF_InsertSortPrio(), and CF_MoveTransaction().

11.55.2.5 suspended bool CF_Flags_Common::suspended

Definition at line 279 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CycleTxFirstActive(), CF_CFDP_DoTick(), CF_CFDP_MsgOutGet(), and CF_DoSuspRes_Txn().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.56 CF_Flags_Rx Struct Reference

Flags that apply to receive transactions.

#include <cf_cfdp_types.h>

Data Fields

- [CF_Flags_Common_t com](#)
- bool [md_recv](#)
md received for r state
- bool [eof_recv](#)
- bool [send_nak](#)
- bool [send_fin](#)
- bool [send_ack](#)
- bool [inactivity_fired](#)
used for r2
- bool [complete](#)
r2
- bool [fd_nak_sent](#)
latches that at least one NAK has been sent for file data

11.56.1 Detailed Description

Flags that apply to receive transactions.

Definition at line 287 of file cf_cfdp_types.h.

11.56.2 Field Documentation

11.56.2.1 com CF_Flags_Common_t CF_Flags_Rx::com

Definition at line 289 of file cf_cfdp_types.h.

11.56.2.2 complete bool CF_Flags_Rx::complete r2

Definition at line 297 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_R_Tick().

11.56.2.3 eof_recv bool CF_Flags_Rx::eof_recv

Definition at line 292 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), and CF_CFDP_R2_SubstateRecvEof().

11.56.2.4 fd_nak_sent bool CF_Flags_Rx::fd_nak_sent

latches that at least one NAK has been sent for file data

Definition at line 298 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_R_SubstateSendNak().

11.56.2.5 inactivity_fired bool CF_Flags_Rx::inactivity_fired

used for r2

Definition at line 296 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R_Tick().

11.56.2.6 md_recv bool CF_Flags_Rx::md_recv

md received for r state

Definition at line 291 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_Init(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_RecvIdle().

11.56.2.7 send_ack bool CF_Flags_Rx::send_ack

Definition at line 295 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_SubstateRecvEof(), and CF_CFDP_R_Tick().

11.56.2.8 send_fin bool CF_Flags_Rx::send_fin

Definition at line 294 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_Cancel(), and CF_CFDP_R_Tick().

11.56.2.9 send_nak bool CF_Flags_Rx::send_nak

Definition at line 293 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), and CF_CFDP_R_Tick().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_cfdp_types.h](#)

11.57 CF_Flags_Tx Struct Reference

Flags that apply to send transactions.

#include <cf_cfdp_types.h>

Data Fields

- [CF_Flags_Common_t](#) com
- bool [md_need_send](#)
- bool [cmd_tx](#)

indicates transaction is commanded (ground) tx

11.57.1 Detailed Description

Flags that apply to send transactions.

Definition at line 304 of file cf_cfdp_types.h.

11.57.2 Field Documentation

11.57.2.1 cmd_tx bool CF_Flags_Tx::cmd_tx

indicates transaction is commanded (ground) tx

Definition at line 309 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ResetTransaction(), and CF_CFDP_TxFile().

11.57.2.2 com [CF_Flags_Common_t](#) CF_Flags_Tx::com

Definition at line 306 of file cf_cfdp_types.h.

11.57.2.3 md_need_send bool CF_Flags_Tx::md_need_send

Definition at line 308 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S2_Nak(), and CF_CFDP_S_CheckAndRespondNak().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_cfdp_types.h](#)

11.58 CF_FreezeCmd Struct Reference

Freeze command structure.

#include <default_cf_msgstruct.h>

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_UnionArgs_Payload_t](#) Payload
Generic command arguments.

11.58.1 Detailed Description

Freeze command structure.

For command details see [CF_FREEZE_CC](#)

Definition at line 115 of file default_cf_msgstruct.h.

11.58.2 Field Documentation

11.58.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_FreezeCmd::CommandHeader

Command header.

Definition at line 117 of file default_cf_msgstruct.h.

11.58.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_FreezeCmd::Payload

Generic command arguments.

Definition at line 118 of file default_cf_msgstruct.h.

Referenced by CF_FreezeCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.59 CF_GapComputeArgs_t Struct Reference

Argument for Gap Compute function.

```
#include <cf_cfdp_r.h>
```

Data Fields

- [CF_Transaction_t](#) * txn
Current transaction being processed.
- [CF_Logical_PduNak_t](#) * nak
Current NAK PDU contents.

11.59.1 Detailed Description

Argument for Gap Compute function.

This is used in conjunction with CF_CFDP_R2_GapCompute

Definition at line 39 of file cf_cfdp_r.h.

11.59.2 Field Documentation

11.59.2.1 nak `CF_Logical_PduNak_t*` `CF_GapComputeArgs_t::nak`
Current NAK PDU contents.
Definition at line 42 of file cf_cfdp_r.h.
Referenced by CF_CFDP_R2_GapCompute().

11.59.2.2 txn `CF_Transaction_t*` `CF_GapComputeArgs_t::txn`
Current transaction being processed.
Definition at line 41 of file cf_cfdp_r.h.
The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_r.h`

11.60 CF_GetParam_Payload Struct Reference

Get parameter command structure.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint8 key`
Parameter key, see [CF_GetSet_ValueID_t](#).
- `uint8 chan_num`
Channel number.

11.60.1 Detailed Description

Get parameter command structure.

For command details see [CF_GET_PARAM_CC](#)

Definition at line 217 of file default_cf_msgdefs.h.

11.60.2 Field Documentation

11.60.2.1 chan_num `uint8 CF_GetParam_Payload::chan_num`
Channel number.

Definition at line 220 of file default_cf_msgdefs.h.

Referenced by CF_GetParamCmd().

11.60.2.2 key `uint8 CF_GetParam_Payload::key`
Parameter key, see [CF_GetSet_ValueID_t](#).

Definition at line 219 of file default_cf_msgdefs.h.

Referenced by CF_GetParamCmd().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.61 CF_GetParamCmd Struct Reference

Get parameter command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_GetParam_Payload_t](#) Payload

11.61.1 Detailed Description

Get parameter command structure.

For command details see [CF_GET_PARAM_CC](#)

Definition at line 192 of file default_cf_msgstruct.h.

11.61.2 Field Documentation

11.61.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_GetParamCmd::CommandHeader

Command header.

Definition at line 194 of file default_cf_msgstruct.h.

11.61.2.2 Payload [CF_GetParam_Payload_t](#) CF_GetParamCmd::Payload

Definition at line 195 of file default_cf_msgstruct.h.

Referenced by CF_GetParamCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgstruct.h

11.62 CF_History Struct Reference

CF History entry.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_TxnFilenames_t](#) fnames
file names associated with this history entry
- [CF_CListNode_t](#) cl_node
for connection to a CList
- [CF_Direction_t](#) dir
direction of this history entry
- [CF_TxnStatus_t](#) txn_stat
final status of operation
- [CF_EntityId_t](#) src_eid
the source eid of the transaction
- [CF_EntityId_t](#) peer_eid
peer_eid is always the "other guy", same src_eid for RX
- [CF_TransactionSeq_t](#) seq_num
transaction identifier, stays constant for entire transfer

11.62.1 Detailed Description

CF History entry.

Records CF app operations for future reference

Definition at line 173 of file cf_cfdp_types.h.

11.62.2 Field Documentation

11.62.2.1 cl_node [CF_CListNode_t](#) CF_History::cl_node

for connection to a CList

Definition at line 176 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitEngine(), CF_CFDP_ResetTransaction(), CF_FindUnusedTransaction(), and CF_ResetHistory().

11.62.2.2 dir [CF_Direction_t](#) CF_History::dir

direction of this history entry

Definition at line 177 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ReceiveMessage(), CF_CFDP_ResetTransaction(), CF_CFDP_SendEotPkt(), CF_CFDP_TxFile_Initiate(), CF_FindUnusedTransaction(), CF_Traverse_WriteHistoryQueueEntryToFile(), and CF_WriteHistoryEntryToFile().

11.62.2.3 fnames [CF_TxnFilenames_t](#) CF_History::fnames

file names associated with this history entry

Definition at line 175 of file cf_cfdp_types.h.

Referenced by CF_CFDP_HandleNotKeepFile(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_Init(), CF_CFDP_RecvMd(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_SendEotPkt(), CF_CFDP_SendMd(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), and CF_WriteHistoryEntryToFile().

11.62.2.4 peer_eid [CF_EntityId_t](#) CF_History::peer_eid

peer_eid is always the "other guy", same src_eid for RX

Definition at line 180 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_RecvIdle(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFile_Initiate(), and CF_WriteHistoryEntryToFile().

11.62.2.5 seq_num [CF_TransactionSeq_t](#) CF_History::seq_num

transaction identifier, stays constant for entire transfer

Definition at line 181 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_RecvIdle(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFile_Initiate(), CF_FindTransactionBySequenceNumber_Impl(), and CF_WriteHistoryEntryToFile().

11.62.2.6 src_eid [CF_EntityId_t](#) CF_History::src_eid

the source eid of the transaction

Definition at line 179 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_RecvIdle(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_SendEotPkt(), CF_CFDP_TxFile_Initiate(), CF_FindTransactionBySequenceNumber_Impl(), and CF_WriteHistoryEntryToFile().

11.62.2.7 txn_stat [CF_TxnStatus_t](#) CF_History::txn_stat

final status of operation

Definition at line 178 of file cf_cfdp_types.h.

Referenced by CF_CFDP_HandleNotKeepFile(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_DispatchRecv(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), CF_CFDP_SetTxnStatus(), and CF_WriteHistoryEntryToFile().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.63 CF_HkChannel_Data Struct Reference

Housekeeping channel data.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- [CF_HkCounters_t](#) counters
Counters.
- [uint16](#) q_size [[CF_QueueIdx_NUM](#)]
Queue sizes.
- [uint8](#) poll_counter
Number of active polling directories.
- [uint8](#) playback_counter
Number of active playback directories.
- [uint8](#) frozen
Frozen state: 0 == not frozen, else frozen.
- [uint8](#) spare [7]
Alignment spare (uint64 values in the counters)

11.63.1 Detailed Description

Housekeeping channel data.

Definition at line 103 of file default_cf_msgdefs.h.

11.63.2 Field Documentation

11.63.2.1 counters `CF_HkCounters_t` `CF_HkChannel_Data::counters`

Counters.

Definition at line 105 of file default_cf_msgdefs.h.

Referenced by `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_R2_CalcCrcChunk()`, `CF_CFDP_R2_Complete()`, `CF_CFDP_R2_Recv_fin_ack()`, `CF_CFDP_R2_RecvMd()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_R_Init()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendInactivityEvent()`, `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_R_SubstateSendNak()`, `CF_CFDP_R_Tick()`, `CF_CFDP_ReceiveMessage()`, `CF_CFDP_RecvDrop()`, `CF_CFDP_RecvFd()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_RecvMd()`, `CF_CFDP_RecvPh()`, `CF_CFDP_S2_Nak()`, `CF_CFDP_S2_WaitForEofAck()`, `CF_CFDP_S_DispatchRecv()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_S_SubstateSendMetadata()`, `CF_CFDP_S_Tick()`, `CF_CFDP_Send()`, and `CF_ResetCountersCmd()`.

11.63.2.2 frozen `uint8` `CF_HkChannel_Data::frozen`

Frozen state: 0 == not frozen, else frozen.

Definition at line 109 of file default_cf_msgdefs.h.

Referenced by `CF_CFDP_CycleEngine()`, `CF_CFDP_MsgOutGet()`, and `CF_DoFreezeThaw()`.

11.63.2.3 playback_counter `uint8` `CF_HkChannel_Data::playback_counter`

Number of active playback directories.

Definition at line 108 of file default_cf_msgdefs.h.

Referenced by `CF_CFDP_ProcessPlaybackDirectories()`.

11.63.2.4 poll_counter `uint8` `CF_HkChannel_Data::poll_counter`

Number of active polling directories.

Definition at line 107 of file default_cf_msgdefs.h.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

11.63.2.5 q_size `uint16` `CF_HkChannel_Data::q_size[CF_QueueIdx_NUM]`

Queue sizes.

Definition at line 106 of file default_cf_msgdefs.h.

Referenced by `CF_CFDP_DisableEngine()`, `CF_CFDP_ReceiveMessage()`, `CF_CList_InsertAfter_Ex()`, `CF_CList_InsertBack_Ex()`, `CF_CList_Remove_Ex()`, `CF_DequeueTransaction()`, and `CF_MoveTransaction()`.

11.63.2.6 spare `uint8` `CF_HkChannel_Data::spare[7]`

Alignment spare (uint64 values in the counters)

Definition at line 110 of file default_cf_msgdefs.h.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.64 CF_HkCmdCounters Struct Reference

Housekeeping command counters.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint16 cmd`

Command success counter.

- `uint16 err`

Command error counter.

11.64.1 Detailed Description

Housekeeping command counters.

Definition at line 39 of file default_cf_msgdefs.h.

11.64.2 Field Documentation

11.64.2.1 cmd `uint16 CF_HkCmdCounters::cmd`

Command success counter.

Definition at line 41 of file default_cf_msgdefs.h.

Referenced by CF_AbandonCmd(), CF_CancelCmd(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DisableEngineCmd(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_NoopCmd(), CF_PlaybackDirCmd(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_ThawCmd(), CF_TxFileCmd(), and CF_WriteQueueCmd().

11.64.2.2 err `uint16 CF_HkCmdCounters::err`

Command error counter.

Definition at line 42 of file default_cf_msgdefs.h.

Referenced by CF_AbandonCmd(), CF_AppPipe(), CF_CancelCmd(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_PlaybackDirCmd(), CF_ProcessGroundCommand(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_ThawCmd(), CF_TxFileCmd(), and CF_WriteQueueCmd().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.65 CF_HkCounters Struct Reference

Housekeeping counters.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `CF_HkSent_t sent`

Sent counters.

- `CF_HkRecv_t recv`

Received counters.

- `CF_HkFault_t fault`

Fault counters.

11.65.1 Detailed Description

Housekeeping counters.

Definition at line 93 of file default_cf_msgdefs.h.

11.65.2 Field Documentation

11.65.2.1 fault [CF_HkFault_t](#) CF_HkCounters::fault

Fault counters.

Definition at line 97 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_Tick(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), and CF_ResetCountersCmd().

11.65.2.2 recv [CF_HkRecv_t](#) CF_HkCounters::recv

Received counters.

Definition at line 96 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvDrop(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), and CF_ResetCountersCmd().

11.65.2.3 sent [CF_HkSent_t](#) CF_HkCounters::sent

Sent counters.

Definition at line 95 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_SubstateSendNak(), CF_CFDP_S_SendFileData(), CF_CFDP_Send(), and CF_ResetCountersCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgdefs.h](#)

11.66 CF_HkFault Struct Reference

Housekeeping fault counters.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- [uint16 file_open](#)
File open fault counter.
- [uint16 file_read](#)
File read fault counter.
- [uint16 file_seek](#)
File seek fault counter.
- [uint16 file_write](#)
File write fault counter.
- [uint16 file_rename](#)
File rename fault counter.
- [uint16 directory_read](#)
Directory read fault counter.
- [uint16 crc_mismatch](#)
CRC mismatch fault counter.
- [uint16 file_size_mismatch](#)
File size mismatch fault counter.
- [uint16 nak_limit](#)
NAK limit exceeded fault counter.

- `uint16 ack_limit`
ACK limit exceeded fault counter.
- `uint16 inactivity_timer`
Inactivity timer exceeded counter.
- `uint16 spare`
Alignment spare to avoid implicit padding.

11.66.1 Detailed Description

Housekeeping fault counters.

Definition at line 74 of file default_cf_msgdefs.h.

11.66.2 Field Documentation

11.66.2.1 `ack_limit uint16 CF_HkFault::ack_limit`

ACK limit exceeded fault counter.

Definition at line 85 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

11.66.2.2 `crc_mismatch uint16 CF_HkFault::crc_mismatch`

CRC mismatch fault counter.

Definition at line 82 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_CheckCrc().

11.66.2.3 `directory_read uint16 CF_HkFault::directory_read`

Directory read fault counter.

Definition at line 81 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_PlaybackDir_Initiate().

11.66.2.4 `file_open uint16 CF_HkFault::file_open`

File open fault counter.

Definition at line 76 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_RecvMd(), CF_CFDP_R_Init(), and CF_CFDP_S_SubstateSendMetadata().

11.66.2.5 `file_read uint16 CF_HkFault::file_read`

File read fault counter.

Definition at line 77 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_S_SendFileData().

11.66.2.6 `file_rename uint16 CF_HkFault::file_rename`

File rename fault counter.

Definition at line 80 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_RecvMd().

11.66.2.7 file_seek uint16 CF_HkFault::file_seek

File seek fault counter.

Definition at line 78 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R_ProcessFd(), CF_CFDP_S_SendFileData(), and CF_CFDP_S_SubstateSendMetadata().

11.66.2.8 file_size_mismatch uint16 CF_HkFault::file_size_mismatch

File size mismatch fault counter.

Definition at line 83 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_RecvMd(), and CF_CFDP_R_SubstateRecvEof().

11.66.2.9 file_write uint16 CF_HkFault::file_write

File write fault counter.

Definition at line 79 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_ProcessFd().

11.66.2.10 inactivity_timer uint16 CF_HkFault::inactivity_timer

Inactivity timer exceeded counter.

Definition at line 86 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_SendInactivityEvent(), and CF_CFDP_S_Tick().

11.66.2.11 nak_limit uint16 CF_HkFault::nak_limit

NAK limit exceeded fault counter.

Definition at line 84 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_Complete().

11.66.2.12 spare uint16 CF_HkFault::spare

Alignment spare to avoid implicit padding.

Definition at line 87 of file default_cf_msgdefs.h.

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgdefs.h

11.67 CF_HkPacket Struct Reference

Housekeeping packet.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CF_HkPacket_Payload_t Payload](#)

11.67.1 Detailed Description

Housekeeping packet.

Definition at line 47 of file default_cf_msgstruct.h.

11.67.2 Field Documentation

11.67.2.1 Payload [CF_HkPacket_Payload_t](#) CF_HkPacket::Payload

Definition at line 50 of file default_cf_msgstruct.h.

Referenced by CF_AbandonCmd(), CF_AppPipe(), CF_CancelCmd(), CF_CFDP_CycleEngine(), CF_CFDP_DisableEngine(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvDrop(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_S2_Nak(), CF_CFD_P_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_Send(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DisableEngineCmd(), CF_DoFreezeThaw(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_MoveTransaction(), CF_NoopCmd(), CF_PlaybackDirCmd(), CF_ProcessGroundCommand(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_ThawCmd(), CF_TxFileCmd(), and CF_WriteQueueCmd().

11.67.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CF_HkPacket::TelemetryHeader

Telemetry header.

Definition at line 49 of file default_cf_msgstruct.h.

Referenced by CF_AppInit(), and CF_SendHkCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgstruct.h

11.68 CF_HkPacket_Payload Struct Reference

Housekeeping packet.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- [CF_HkCmdCounters_t](#) counters
Command counters.
- [uint8](#) spare [4]
Alignment spare (CF_HkCmdCounters_t is 4 bytes)
- [CF_HkChannel_Data_t](#) channel_hk [[CF_NUM_CHANNELS](#)]
Per channel housekeeping data.

11.68.1 Detailed Description

Housekeeping packet.

Definition at line 116 of file default_cf_msgdefs.h.

11.68.2 Field Documentation

11.68.2.1 channel_hk `CF_HkChannel_Data_t` `CF_HkPacket_Payload::channel_hk[CF_NUM_CHANNELS]`

Per channel housekeeping data.

Definition at line 121 of file `default_cf_msgdefs.h`.

Referenced by `CF_CFDP_CycleEngine()`, `CF_CFDP_DisableEngine()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_ProcessPlaybackDirectories()`, `CF_CFDP_ProcessPollingDirectories()`, `CF_CFDP_R2_CalcCrcChunk()`, `CF_CFDP_R2_Complete()`, `CF_CFDP_R2_Recv_fin_ack()`, `CF_CFDP_R2_RecvMd()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_R_Init()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendInactivityEvent()`, `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_R_SubstateSendNak()`, `CF_CFDP_R_Tick()`, `CF_CFDP_ReceiveMessage()`, `CF_CFDP_RecvDrop()`, `CF_CFDP_RecvFd()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_RecvMd()`, `CF_CFDP_RecvPh()`, `CF_CFDP_S2_Nak()`, `CF_CFDP_S2_WaitForEofAck()`, `CF_CFDP_S_DispatchRecv()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_S_SubstateSendMetadata()`, `CF_CFDP_S_Tick()`, `CF_CFDP_Send()`, `CF_CList_InsertAfter_Ex()`, `CF_CList_InsertBack_Ex()`, `CF_CList_Remove_Ex()`, `CF_DequeueTransaction()`, `CF_DoFreezeThaw()`, `CF_MoveTransaction()`, and `CF_ResetCountersCmd()`.

11.68.2.2 counters `CF_HkCmdCounters_t` `CF_HkPacket_Payload::counters`

Command counters.

Definition at line 118 of file `default_cf_msgdefs.h`.

Referenced by `CF_AbandonCmd()`, `CF_AppPipe()`, `CF_CancelCmd()`, `CF_DisableDequeueCmd()`, `CF_DisableDirPollingCmd()`, `CF_DisableEngineCmd()`, `CF_DoSuspRes()`, `CF_EnableDequeueCmd()`, `CF_EnableDirPollingCmd()`, `CF_EnableEngineCmd()`, `CF_FreezeCmd()`, `CF_SetParamCmd()`, `CF_NoopCmd()`, `CF_PlaybackDirCmd()`, `CF_ProcessGroundCommand()`, `CF_PurgeQueueCmd()`, `CF_ResetCountersCmd()`, `CF_ThawCmd()`, `CF_TxFileCmd()`, and `CF_WriteQueueCmd()`.

11.68.2.3 spare `uint8` `CF_HkPacket_Payload::spare[4]`

Alignment spare (`CF_HkCmdCounters_t` is 4 bytes)

Definition at line 119 of file `default_cf_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.69 CF_HkRecv Struct Reference

Housekeeping received counters.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint64 file_data_bytes`
Received File data bytes.
- `uint32 pdu`
Received PDUs with valid header counter.
- `uint32 error`
Received PDUs with error counter, see related event for cause.
- `uint16 spurious`
Received PDUs with invalid directive code for current context or file directive FIN without matching active transaction counter, see related event for cause.
- `uint16 dropped`
Received PDUs dropped due to a transaction error.
- `uint32 nak_segment_requests`
Received NAK segment requests counter.

11.69.1 Detailed Description

Housekeeping received counters.

Definition at line 58 of file default_cf_msgdefs.h.

11.69.2 Field Documentation

11.69.2.1 dropped `uint16 CF_HkRecv::dropped`

Received PDUs dropped due to a transaction error.

Definition at line 67 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_DispatchRecv(), and CF_CFDP_RecvDrop().

11.69.2.2 error `uint32 CF_HkRecv::error`

Received PDUs with error counter, see related event for cause.

Definition at line 62 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_S2_Nak(), and CF_CFDP_S2_WaitForEofAck().

11.69.2.3 file_data_bytes `uint64 CF_HkRecv::file_data_bytes`

Received File data bytes.

Definition at line 60 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_ProcessFd().

11.69.2.4 nak_segment_requests `uint32 CF_HkRecv::nak_segment_requests`

Received NAK segment requests counter.

Definition at line 68 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_S2_Nak().

11.69.2.5 pdu `uint32 CF_HkRecv::pdu`

Received PDUs with valid header counter.

Definition at line 61 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_RecvPh().

11.69.2.6 spurious `uint16 CF_HkRecv::spurious`

Received PDUs with invalid directive code for current context or file directive FIN without matching active transaction counter, see related event for cause.

Definition at line 63 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_DispatchRecv(), CF_CFDP_ReceiveMessage(), and CF_CFDP_S_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgdefs.h](#)

11.70 CF_HkSent Struct Reference

Housekeeping sent counters.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint64 file_data_bytes`
Sent File data bytes.
- `uint32 pdu`
Sent PDUs counter.
- `uint32 nak_segment_requests`
Sent NAK segment requests counter.

11.70.1 Detailed Description

Housekeeping sent counters.

Definition at line 48 of file default_cf_msgdefs.h.

11.70.2 Field Documentation

11.70.2.1 `file_data_bytes` `uint64` CF_HkSent::file_data_bytes

Sent File data bytes.

Definition at line 50 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_S_SendFileData().

11.70.2.2 `nak_segment_requests` `uint32` CF_HkSent::nak_segment_requests

Sent NAK segment requests counter.

Definition at line 52 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_R_SubstateSendNak().

11.70.2.3 `pdu` `uint32` CF_HkSent::pdu

Sent PDUs counter.

Definition at line 51 of file default_cf_msgdefs.h.

Referenced by CF_CFDP_Send().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.71 CF_Input Struct Reference

CF engine input state.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `CFE_SB_Buffer_t * msg`
Binary message received from underlying transport.
- `CF_DecoderState_t decode`
Decoding state (while interpreting message)
- `CF_Logical_PduBuffer_t rx_pdudata`
Rx PDU logical values.

11.71.1 Detailed Description

CF engine input state.
Keeps the state of the current input PDU in the CF engine
Definition at line 430 of file cf_cfdp_types.h.

11.71.2 Field Documentation

11.71.2.1 decode `CF_DecoderState_t CF_Input::decode`

Decoding state (while interpreting message)
Definition at line 433 of file cf_cfdp_types.h.
Referenced by CF_CFDP_ReceiveMessage().

11.71.2.2 msg `CFE_SB_Buffer_t* CF_Input::msg`

Binary message received from underlying transport.
Definition at line 432 of file cf_cfdp_types.h.

11.71.2.3 rx_pdudata `CF_Logical_PduBuffer_t CF_Input::rx_pdudata`

Rx PDU logical values.
Definition at line 434 of file cf_cfdp_types.h.
Referenced by CF_CFDP_ReceiveMessage().
The documentation for this struct was generated from the following file:

- apps/cf/fsd/src/[cf_cfdp_types.h](#)

11.72 CF_Logical_IntHeader Union Reference

A union of all possible internal header types in a PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `CF_Logical_PduEof_t eof`
valid when pdu_type=0 + directive_code=EOF (4)
- `CF_Logical_PduFin_t fin`
valid when pdu_type=0 + directive_code=FIN (5)
- `CF_Logical_PduAck_t ack`
valid when pdu_type=0 + directive_code=ACK (6)
- `CF_Logical_PduMd_t md`
valid when pdu_type=0 + directive_code=METADATA (7)
- `CF_Logical_PduNak_t nak`
valid when pdu_type=0 + directive_code=NAK (8)
- `CF_Logical_PduFileDataHeader_t fd`
valid when pdu_type=1 (directive_code is not applicable)

11.72.1 Detailed Description

A union of all possible internal header types in a PDU.
The specific entry which applies depends on the combination of pdu type and directive code.
Definition at line 312 of file cf_logical_pdu.h.

11.72.2 Field Documentation

11.72.2.1 ack `CF_Logical_PduAck_t` CF_Logical_IntHeader::ack

valid when pdu_type=0 + directive_code=ACK (6)

Definition at line 316 of file cf_logical_pdu.h.

Referenced by CF_CFDP_RecvAck(), and CF_CFDP_SendAck().

11.72.2.2 eof `CF_Logical_PduEof_t` CF_Logical_IntHeader::eof

valid when pdu_type=0 + directive_code=EOF (4)

Definition at line 314 of file cf_logical_pdu.h.

Referenced by CF_CFDP_R1_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_RecvEof(), and CF_CFDP_SendEof().

11.72.2.3 fd `CF_Logical_PduFileDataHeader_t` CF_Logical_IntHeader::fd

valid when pdu_type=1 (directive_code is not applicable)

Definition at line 319 of file cf_logical_pdu.h.

Referenced by CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_ProcessFd(), CF_CFDP_RecvFd(), and CF_CFDP_S_SendFileData().

11.72.2.4 fin `CF_Logical_PduFin_t` CF_Logical_IntHeader::fin

valid when pdu_type=0 + directive_code=FIN (5)

Definition at line 315 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ReceiveMessage(), CF_CFDP_RecvFin(), CF_CFDP_S2_Fin(), and CF_CFDP_SendFin().

11.72.2.5 md `CF_Logical_PduMd_t` CF_Logical_IntHeader::md

valid when pdu_type=0 + directive_code=METADATA (7)

Definition at line 317 of file cf_logical_pdu.h.

Referenced by CF_CFDP_RecvMd(), and CF_CFDP_SendMd().

11.72.2.6 nak `CF_Logical_PduNak_t` CF_Logical_IntHeader::nak

valid when pdu_type=0 + directive_code=NAK (8)

Definition at line 318 of file cf_logical_pdu.h.

Referenced by CF_CFDP_R_SubstateSendNak(), CF_CFDP_RecvNak(), CF_CFDP_S2_Nak(), and CF_CFDP_SendNak().

The documentation for this union was generated from the following file:

- [apps/cf/fsd/src/cf_logical_pdu.h](#)

11.73 CF_Logical_Lv Struct Reference

Structure representing logical LV Object format.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `uint8 length`

Length of data field.

- `const void * data_ptr`

Source of actual data in original location.

11.73.1 Detailed Description

Structure representing logical LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

These are only used for string data (mostly filenames) so the data can refer directly to the encoded bits, it does not necessarily need to be duplicated here.

Definition at line 144 of file cf_logical_pdu.h.

11.73.2 Field Documentation

11.73.2.1 `data_ptr const void* CF_Logical_Lv::data_ptr`

Source of actual data in original location.

Definition at line 147 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_CopyStringFromLV()`, `CF_CFDP_DecodeLV()`, `CF_CFDP_EncodeLV()`, and `CF_CFDP_SendMd()`.

11.73.2.2 `length uint8 CF_Logical_Lv::length`

Length of data field.

Definition at line 146 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_CopyStringFromLV()`, `CF_CFDP_DecodeLV()`, `CF_CFDP_EncodeLV()`, `CF_CFDP_RecvMd()`, and `CF_CFDP_SendMd()`.

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_logical_pdu.h](#)

11.74 CF_Logical_PduAck Struct Reference

Structure representing CFDP Acknowledge PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `uint8 ack_directive_code`
directive code of the PDU being ACK'ed
- `uint8 ack_subtype_code`
depends on ack_directive_code
- `CF_CFDP_ConditionCode_t cc`
- `CF_CFDP_AckTxnStatus_t txn_status`

11.74.1 Detailed Description

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Definition at line 252 of file cf_logical_pdu.h.

11.74.2 Field Documentation

11.74.2.1 ack_directive_code `uint8 CF_Logical_PduAck::ack_directive_code`
directive code of the PDU being ACK'ed

Definition at line 254 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAck(), CF_CFDP_EncodeAck(), and CF_CFDP_SendAck().

11.74.2.2 ack_subtype_code `uint8 CF_Logical_PduAck::ack_subtype_code`
depends on ack_directive_code

Definition at line 255 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAck(), CF_CFDP_EncodeAck(), and CF_CFDP_SendAck().

11.74.2.3 cc `CF_CFDP_ConditionCode_t CF_Logical_PduAck::cc`

Definition at line 256 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAck(), CF_CFDP_EncodeAck(), and CF_CFDP_SendAck().

11.74.2.4 txn_status `CF_CFDP_AckTxnStatus_t CF_Logical_PduAck::txn_status`

Definition at line 257 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAck(), CF_CFDP_EncodeAck(), and CF_CFDP_SendAck().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_logical_pdu.h](#)

11.75 CF_Logical_PduBuffer Struct Reference

Encapsulates the entire PDU information.

```
#include <cf_logical_pdu.h>
```

Data Fields

- struct [CF_EncoderState](#) * penc
- struct [CF_DecoderState](#) * pdec
- [CF_Logical_PduHeader_t](#) pdu_header

Data in PDU header is applicable to all packets.

- [CF_Logical_PduFileDirectiveHeader_t](#) fdirective

The directive code applies to file directive PDUs, where the pdu_type in the common header is 0. Otherwise this value should be set to 0 for data PDUs (which is a reserved value and does not alias any valid directive code).

- [CF_Logical_IntHeader_t](#) int_header

The internal header is specific to the type of PDU being processed. This is a union of all those possible types. See the union definition for which member applies to a given processing cycle.

- `uint32 content_crc`

Some PDU types might have a CRC at the end. If so, this field reflects the value of that CRC. Its presence/validity depends on the pdu_type and crc_flag in the pdu_header.

11.75.1 Detailed Description

Encapsulates the entire PDU information.

Definition at line 326 of file cf_logical_pdu.h.

11.75.2 Field Documentation

11.75.2.1 content_crc `uint32 CF_Logical_PduBuffer::content_crc`

Some PDU types might have a CRC at the end. If so, this field reflects the value of that CRC. Its presence/validity depends on the pdu_type and crc_flag in the pdu_header.

Note that all CFDP CRCs are 32 bits in length, the blue book does not permit for any other size.

Definition at line 365 of file cf_logical_pdu.h.

11.75.2.2 fdirective `CF_Logical_PduFileDirectiveHeader_t CF_Logical_PduBuffer::fdirective`

The directive code applies to file directive PDUs, where the pdu_type in the common header is 0. Otherwise this value should be set to 0 for data PDUs (which is a reserved value and does not alias any valid directive code).

Definition at line 347 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ConstructPduHeader(), CF_CFDP_R_DispatchRecv(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_RecvPh(), and CF_CFDP_S_DispatchRecv().

11.75.2.3 int_header `CF_Logical_IntHeader_t CF_Logical_PduBuffer::int_header`

The internal header is specific to the type of PDU being processed. This is a union of all those possible types. See the union definition for which member applies to a given processing cycle.

Definition at line 355 of file cf_logical_pdu.h.

Referenced by CF_CFDP_R1_SubstateRecvEof(), CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvAck(), CF_CFDP_RecvEof(), CF_CFDP_RecvFd(), CF_CFDP_RecvFin(), CF_CFDP_RecvMd(), CF_CFDP_RecvNak(), CF_CFDP_S2_Fin(), CF_CFDP_S2_Nak(), CF_CFDP_S_SendFileData(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), and CF_CFDP_SendNak().

11.75.2.4 pdec `struct CF_DecoderState* CF_Logical_PduBuffer::pdec`

Definition at line 334 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeStart(), CF_CFDP_RecvAck(), CF_CFDP_RecvEof(), CF_CFDP_RecvFd(), CF_CFDP_RecvFin(), CF_CFDP_RecvMd(), CF_CFDP_RecvNak(), and CF_CFDP_RecvPh().

11.75.2.5 pdu_header `CF_Logical_PduHeader_t CF_Logical_PduBuffer::pdu_header`

Data in PDU header is applicable to all packets.

Definition at line 339 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ConstructPduHeader(), CF_CFDP_R_DispatchRecv(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvPh(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_Send(), and CF_CFDP_SetPduLength().

11.75.2.6 penc `struct CF_EncoderState* CF_Logical_PduBuffer::penc`

Definition at line 333 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeStart(), CF_CFDP_S_SendFileData(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_SendNak(), and CF_CFDP_SetPduLength().

The documentation for this struct was generated from the following file:

- apps/cf/fsd/src/[cf_logical_pdu.h](#)

11.76 CF_Logical_PduEof Struct Reference

Structure representing logical End of file PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `CF_CFDP_ConditionCode_t cc`
- `uint32 crc`
- `CF_FileSize_t size`
- `CF_Logical_TlvList_t tlv_list`

Set of all TLV blobs in this PDU.

11.76.1 Detailed Description

Structure representing logical End of file PDU.

See also

[CF_CFDP_PduEof_t](#) for encoded form

Definition at line 218 of file cf_logical_pdu.h.

11.76.2 Field Documentation

11.76.2.1 cc `CF_CFDP_ConditionCode_t CF_Logical_PduEof::cc`

Definition at line 220 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeEof()`, `CF_CFDP_EncodeEof()`, `CF_CFDP_R2_SubstateRecvEof()`, and `CF_CFDP_SendEof()`.

11.76.2.2 crc `uint32 CF_Logical_PduEof::crc`

Definition at line 221 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeEof()`, `CF_CFDP_EncodeEof()`, `CF_CFDP_R1_SubstateRecvEof()`, `CF_CFDP_R2_SubstateRecvEof()`, and `CF_CFDP_SendEof()`.

11.76.2.3 size `CF_FileSize_t CF_Logical_PduEof::size`

Definition at line 222 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeEof()`, `CF_CFDP_EncodeEof()`, `CF_CFDP_R2_SubstateRecvEof()`, `CF_CFDP_R_SubstateRecvEof()`, and `CF_CFDP_SendEof()`.

11.76.2.4 tlv_list `CF_Logical_TlvList_t CF_Logical_PduEof::tlv_list`

Set of all TLV blobs in this PDU.

Definition at line 227 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeEof()`, `CF_CFDP_EncodeEof()`, and `CF_CFDP_SendEof()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsu/src/cf_logical_pdu.h`

11.77 CF_Logical_PduFileDataHeader Struct Reference

```
#include <cf_logical_pdu.h>
```

Data Fields

- `uint8 continuation_state`
- `CF_Logical_SegmentList_t segment_list`

the segment_meta_length value will be stored in the segment_list.num_segments field below
- `CF_FileSize_t offset`

Offset of data in file.
- `const void * data_ptr`

pointer to read-only data blob within encoded PDU
- `size_t data_len`

Length of data blob within encoded PDU (derived field)

11.77.1 Detailed Description

Definition at line 290 of file cf_logical_pdu.h.

11.77.2 Field Documentation

11.77.2.1 `continuation_state` `uint8 CF_Logical_PduFileDataHeader::continuation_state`

Definition at line 292 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, and `CF_CFDP_EncodeFileDataHeader()`.

11.77.2.2 `data_len` `size_t CF_Logical_PduFileDataHeader::data_len`

Length of data blob within encoded PDU (derived field)

Definition at line 303 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_R1_SubstateRecvFileData()`, `CF_CFDP_R2_SubstateRecvFileData()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_RecvFd()`, and `CF_CFDP_S_SendFileData()`.

11.77.2.3 `data_ptr` `const void* CF_Logical_PduFileDataHeader::data_ptr`

pointer to read-only data blob within encoded PDU

Definition at line 302 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_R1_SubstateRecvFileData()`, `CF_CFDP_R_ProcessFd()`, and `CF_CFDP_S_SendFileData()`.

11.77.2.4 `offset` `CF_FileSize_t CF_Logical_PduFileDataHeader::offset`

Offset of data in file.

Definition at line 300 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_EncodeFileDataHeader()`, `CF_CFDP_R2_SubstateRecvFileData()`, `CF_CFDP_R_ProcessFd()`, and `CF_CFDP_S_SendFileData()`.

11.77.2.5 `segment_list` `CF_Logical_SegmentList_t CF_Logical_PduFileDataHeader::segment_list`

the segment_meta_length value will be stored in the segment_list.num_segments field below

Definition at line 298 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, and `CF_CFDP_EncodeFileDataHeader()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_logical_pdu.h`

11.78 CF_Logical_PduFileDirectiveHeader Struct Reference

Structure representing logical File Directive header.

```
#include <cf_logical_pdu.h>
```

Data Fields

- [CF_CFDP_FileDirective_t directive_code](#)

11.78.1 Detailed Description

Structure representing logical File Directive header.

This contains the file directive code from the PDUs for which it applies. The codes are mapped directly to the CFDP protocol values, but converted to a native value (enum) for direct use by software.

Definition at line 129 of file cf_logical_pdu.h.

11.78.2 Field Documentation

11.78.2.1 directive_code [CF_CFDP_FileDirective_t](#) CF_Logical_PduFileDirectiveHeader::directive_code

Definition at line 131 of file cf_logical_pdu.h.

Referenced by [CF_CFDP_ConstructPduHeader\(\)](#), [CF_CFDP_DecodeFileDirectiveHeader\(\)](#), [CF_CFDP_EncodeFileDirectiveHeader\(\)](#), [CF_CFDP_R_DispatchRecv\(\)](#), [CF_CFDP_ReceiveMessage\(\)](#), [CF_CFDP_RecvIdle\(\)](#), and [CF_CFDP_S_DispatchRecv\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf_logical_pdu.h](#)

11.79 CF_Logical_PduFin Struct Reference

Structure representing logical Finished PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- [CF_CFDP_ConditionCode_t cc](#)
complete file indicated by '0'. Nonzero means incomplete.
- [CF_CFDP_FinFileStatus_t file_status](#)
- [uint8 delivery_code](#)
Set of all TLV blobs in this PDU.

11.79.1 Detailed Description

Structure representing logical Finished PDU.

See also

[CF_CFDP_PduFin_t](#) for encoded form

Definition at line 235 of file cf_logical_pdu.h.

11.79.2 Field Documentation

11.79.2.1 cc CF_CFDP_ConditionCode_t CF_Logical_PduFin::cc

Definition at line 237 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeFin(), CF_CFDP_EncodeFin(), CF_CFDP_ReceiveMessage(), CF_CFDP_S2_Fin(), and CF_CFDP_SendFin().

11.79.2.2 delivery_code uint8 CF_Logical_PduFin::delivery_code

complete file indicated by '0'. Nonzero means incomplete.

Definition at line 239 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeFin(), CF_CFDP_EncodeFin(), and CF_CFDP_SendFin().

11.79.2.3 file_status CF_CFDP_FinFileStatus_t CF_Logical_PduFin::file_status

Definition at line 238 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeFin(), CF_CFDP_EncodeFin(), and CF_CFDP_SendFin().

11.79.2.4 tlv_list CF_Logical_TlvList_t CF_Logical_PduFin::tlv_list

Set of all TLV blobs in this PDU.

Definition at line 244 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeFin(), CF_CFDP_EncodeFin(), and CF_CFDP_SendFin().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_logical_pdu.h](#)

11.80 CF_Logical_PduHeader Struct Reference

Structure representing base CFDP PDU header.

```
#include <cf_logical_pdu.h>
```

Data Fields

- **uint8 version**
Version of the protocol.
- **uint8 pdu_type**
File Directive (0) or File Data (1)
- **uint8 direction**
Toward Receiver (0) or Toward Sender (1)
- **uint8 txm_mode**
Acknowledged (0) or Unacknowledged (1)
- **uint8 crc_flag**
CRC not present (0) or CRC present (1)
- **uint8 large_flag**
Small/32-bit size (0) or Large/64-bit size (1)
- **uint8 segment_meta_flag**
Segment Metatdata not present (0) or Present (1)
- **uint8 eid_length**
Length of encoded entity IDs, in octets (NOT size of logical value)
- **uint8 txn_seq_length**
Length of encoded sequence number, in octets (NOT size of logical value)
- **uint16 header_encoded_length**

Length of the encoded PDU header, in octets (NOT sizeof struct)

- `uint16 data_encoded_length`

Length of the encoded PDU data, in octets.

- `CF_EntityId_t source_eid`

Source entity ID (normalized)

- `CF_EntityId_t destination_eid`

Destination entity ID (normalized)

- `CF_TransactionSeq_t sequence_num`

Sequence number (normalized)

11.80.1 Detailed Description

Structure representing base CFDP PDU header.

Reflects the common content at the beginning of all CFDP PDUs, of all types.

See also

[CF_CFDP_PduHeader_t](#) for encoded form

Definition at line 101 of file cf_logical_pdu.h.

11.80.2 Field Documentation

11.80.2.1 `crc_flag uint8 CF_Logical_PduHeader::crc_flag`

CRC not present (0) or CRC present (1)

Definition at line 107 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_RecvFd()`.

11.80.2.2 `data_encoded_length uint16 CF_Logical_PduHeader::data_encoded_length`

Length of the encoded PDU data, in octets.

Definition at line 115 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderFinalSize()`, `CF_CFDP_Send()`, and `CF_CFDP_SetPduLength()`.

11.80.2.3 `destination_eid CF_EntityId_t CF_Logical_PduHeader::destination_eid`

Destination entity ID (normalized)

Definition at line 118 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, and `CF_CFDP_ReceiveMessage()`.

11.80.2.4 `direction uint8 CF_Logical_PduHeader::direction`

Toward Receiver (0) or Toward Sender (1)

Definition at line 105 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

11.80.2.5 eid_length `uint8 CF_Logical_PduHeader::eid_length`

Length of encoded entity IDs, in octets (NOT size of logical value)

Definition at line 111 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

11.80.2.6 header_encoded_length `uint16 CF_Logical_PduHeader::header_encoded_length`

Length of the encoded PDU header, in octets (NOT sizeof struct)

Definition at line 114 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_Send()`, and `CF_CFDP_SetPduLength()`.

11.80.2.7 large_flag `uint8 CF_Logical_PduHeader::large_flag`

Small/32-bit size (0) or Large/64-bit size (1)

Definition at line 108 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_RecvPh()`.

11.80.2.8 pdu_type `uint8 CF_Logical_PduHeader::pdu_type`

File Directive (0) or File Data (1)

Definition at line 104 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_RecvPh()`, and `CF_CFDP_S_DispatchRecv()`.

11.80.2.9 segment_meta_flag `uint8 CF_Logical_PduHeader::segment_meta_flag`

Segment Metatdata not present (0) or Present (1)

Definition at line 110 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_RecvFd()`, and `CF_CFDP_S_SendFileData()`.

11.80.2.10 sequence_num `CF_TransactionSeq_t CF_Logical_PduHeader::sequence_num`

Sequence number (normalized)

Definition at line 119 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_ReceiveMessage()`, and `CF_CFDP_RecvIdle()`.

11.80.2.11 source_eid `CF_EntityId_t CF_Logical_PduHeader::source_eid`

Source entity ID (normalized)

Definition at line 117 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_ReceiveMessage()`, and `CF_CFDP_RecvIdle()`.

11.80.2.12 txm_mode `uint8 CF_Logical_PduHeader::txm_mode`

Acknowledged (0) or Unacknowledged (1)

Definition at line 106 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, and `CF_CFDP_RecvIdle()`.

11.80.2.13 txn_seq_length uint8 CF_Logical_PduHeader::txn_seq_length

Length of encoded sequence number, in octets (NOT size of logical value)

Definition at line 112 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ConstructPduHeader(), CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

11.80.2.14 version uint8 CF_Logical_PduHeader::version

Version of the protocol.

Definition at line 103 of file cf_logical_pdu.h.

Referenced by CF_CFDP_ConstructPduHeader(), CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_logical_pdu.h](#)

11.81 CF_Logical_PduMd Struct Reference

Structure representing CFDP Metadata PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- [uint8 close_req](#)
transaction closure not requested (0) or requested (1)
- [uint8 checksum_type](#)
0 indicates legacy modular checksum
- [CF_FileSize_t size](#)
- [CF_Logical_Lv_t source_filename](#)
- [CF_Logical_Lv_t dest_filename](#)

11.81.1 Detailed Description

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

Definition at line 265 of file cf_logical_pdu.h.

11.81.2 Field Documentation

11.81.2.1 checksum_type uint8 CF_Logical_PduMd::checksum_type

0 indicates legacy modular checksum

Definition at line 268 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

11.81.2.2 close_req uint8 CF_Logical_PduMd::close_req

transaction closure not requested (0) or requested (1)

Definition at line 267 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

11.81.2.3 dest_filename `CF_Logical_Lv_t` CF_Logical_PduMd::dest_filename

Definition at line 273 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeMd(), CF_CFDP_EncodeMd(), CF_CFDP_RecvMd(), and CF_CFDP_SendMd().

11.81.2.4 size `CF_FileSize_t` CF_Logical_PduMd::size

Definition at line 270 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeMd(), CF_CFDP_EncodeMd(), CF_CFDP_RecvMd(), and CF_CFDP_SendMd().

11.81.2.5 source_filename `CF_Logical_Lv_t` CF_Logical_PduMd::source_filename

Definition at line 272 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeMd(), CF_CFDP_EncodeMd(), CF_CFDP_RecvMd(), and CF_CFDP_SendMd().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_logical_pdu.h](#)

11.82 CF_Logical_PduNak Struct Reference

Structure representing logical Non-Acknowledge PDU.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `CF_FileSize_t scope_start`
- `CF_FileSize_t scope_end`
- `CF_Logical_SegmentList_t segment_list`

Set of all segments in this PDU.

11.82.1 Detailed Description

Structure representing logical Non-Acknowledge PDU.

Definition at line 279 of file cf_logical_pdu.h.

11.82.2 Field Documentation

11.82.2.1 scope_end `CF_FileSize_t` CF_Logical_PduNak::scope_end

Definition at line 282 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeNak(), CF_CFDP_EncodeNak(), and CF_CFDP_R_SubstateSendNak().

11.82.2.2 scope_start `CF_FileSize_t` CF_Logical_PduNak::scope_start

Definition at line 281 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeNak(), CF_CFDP_EncodeNak(), CF_CFDP_R2_GapCompute(), and CF_CFDP_R_SubstateSendNak().

11.82.2.3 segment_list [CF_Logical_SegmentList_t](#) CF_Logical_PduNak::segment_list

Set of all segments in this PDU.

Definition at line 287 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeNak(), CF_CFDP_EncodeNak(), CF_CFDP_R2_GapCompute(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_S2_Nak().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf_logical_pdu.h\)](#)

11.83 CF_Logical_SegmentList Struct Reference

```
#include <cf_logical_pdu.h>
```

Data Fields

- [uint8 num_segments](#)
number of valid entries in the segment list
- [CF_Logical_SegmentRequest_t segments \[CF_PDU_MAX_SEGMENTS\]](#)
Set of all segment requests in this PDU.

11.83.1 Detailed Description

Definition at line 193 of file cf_logical_pdu.h.

11.83.2 Field Documentation

11.83.2.1 num_segments

[uint8 CF_Logical_SegmentList::num_segments](#)

number of valid entries in the segment list

Definition at line 195 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAllSegments(), CF_CFDP_DecodeFileDataHeader(), CF_CFDP_EncodeAllSegments(), CF_CFDP_EncodeFileDataHeader(), CF_CFDP_R2_GapCompute(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_S2_Nak().

11.83.2.2 segments

[CF_Logical_SegmentRequest_t CF_Logical_SegmentList::segments \[CF_PDU_MAX_SEGMENTS\]](#)

Set of all segment requests in this PDU.

Number of valid entries is indicated by num_segments, and may be 0 if the PDU does not contain any such fields.

Definition at line 203 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeAllSegments(), CF_CFDP_DecodeFileDataHeader(), CF_CFDP_EncodeAllSegments(), CF_CFDP_R2_GapCompute(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_S2_Nak().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf_logical_pdu.h\)](#)

11.84 CF_Logical_SegmentRequest Struct Reference

Structure representing logical Segment Request data.

```
#include <cf_logical_pdu.h>
```

Data Fields

- [CF_FileSize_t offset_start](#)
- [CF_FileSize_t offset_end](#)

11.84.1 Detailed Description

Structure representing logical Segment Request data.
Definition at line 187 of file cf_logical_pdu.h.

11.84.2 Field Documentation

11.84.2.1 offset_end `CF_FileSize_t` CF_Logical_SegmentRequest::offset_end

Definition at line 190 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeSegmentRequest(), CF_CFDP_EncodeSegmentRequest(), CF_CFDP_R2_GapCompute(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_S2_Nak().

11.84.2.2 offset_start `CF_FileSize_t` CF_Logical_SegmentRequest::offset_start

Definition at line 189 of file cf_logical_pdu.h.

Referenced by CF_CFDP_DecodeSegmentRequest(), CF_CFDP_EncodeSegmentRequest(), CF_CFDP_R2_GapCompute(), CF_CFDP_R_SubstateSendNak(), and CF_CFDP_S2_Nak().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_logical_pdu.h](#)

11.85 CF_Logical_Tlv Struct Reference

Structure representing logical TLV Object format.

```
#include <cf_logical_pdu.h>
```

Data Fields

- `CF_CFDP_TlvType_t type`
Nature of data field.
- `uint8 length`
Length of data field (encoded length, not local storage size)
- `CF_Logical_TlvData_t data`

11.85.1 Detailed Description

Structure representing logical TLV Object format.

In the current implementation of CF, only entity IDs are currently encoded in this form where indicated in the spec. This may change in a future version.

See also

[CF_CFDP_tlv_t](#) for encoded form

Definition at line 177 of file cf_logical_pdu.h.

11.85.2 Field Documentation

11.85.2.1 data `CF_Logical_TlvData_t` CF_Logical_Tlv::data

Definition at line 181 of file cf_logical_pdu.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

11.85.2.2 length uint8 CF_Logical_Tlv::length

Length of data field (encoded length, not local storage size)

Definition at line 180 of file cf_logical_pdu.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

11.85.2.3 type CF_CFDP_TlvType_t CF_Logical_Tlv::type

Nature of data field.

Definition at line 179 of file cf_logical_pdu.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

The documentation for this struct was generated from the following file:

- apps/cf/fsd/src/[cf_logical_pdu.h](#)

11.86 CF_Logical_TlvData Union Reference

Union of various data items that may occur in a TLV item.

```
#include <cf_logical_pdu.h>
```

Data Fields

- [CF_EntityId_t eid](#)
Valid when type=ENTITY_ID (6)
- const void * [data_ptr](#)
Source of actual data in original location (other string/binary types)

11.86.1 Detailed Description

Union of various data items that may occur in a TLV item.

The actual type is identified by the "type" field in the enclosing TLV

Currently filestore requests are not implemented in CF, so the TLV use is limited. This may change in the future.

Numeric data needs to actually be copied to this buffer, because it needs to be normalized in length and byte-order. But string data (e.g. filenames, messages) can reside in the original encoded form.

Definition at line 162 of file cf_logical_pdu.h.

11.86.2 Field Documentation

11.86.2.1 data_ptr const void* CF_Logical_TlvData::data_ptr

Source of actual data in original location (other string/binary types)

Definition at line 165 of file cf_logical_pdu.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

11.86.2.2 eid CF_EntityId_t CF_Logical_TlvData::eid

Valid when type=ENTITY_ID (6)

Definition at line 164 of file cf_logical_pdu.h.

Referenced by CF_CFDP_AppendTlv(), CF_CFDP_DecodeTLV(), and CF_CFDP_EncodeTLV().

The documentation for this union was generated from the following file:

- apps/cf/fsd/src/[cf_logical_pdu.h](#)

11.87 CF_Logical_TlvList Struct Reference

```
#include <cf_logical_pdu.h>
```

Data Fields

- `uint8 num_tlv`
number of valid entries in the TLV list
- `CF_Logical_Tlv_t tlv[CF_PDU_MAX_TLV]`

11.87.1 Detailed Description

Definition at line 206 of file cf_logical_pdu.h.

11.87.2 Field Documentation

11.87.2.1 num_tlv `uint8 CF_Logical_TlvList::num_tlv`

number of valid entries in the TLV list

Definition at line 208 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_AppendTlv()`, `CF_CFDP_DecodeAllTlv()`, and `CF_CFDP_EncodeAllTlv()`.

11.87.2.2 tlv `CF_Logical_Tlv_t CF_Logical_TlvList::tlv[CF_PDU_MAX_TLV]`

Definition at line 210 of file cf_logical_pdu.h.

Referenced by `CF_CFDP_AppendTlv()`, `CF_CFDP_DecodeAllTlv()`, and `CF_CFDP_EncodeAllTlv()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_logical_pdu.h`

11.88 CF_NoopCmd Struct Reference

Noop command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.88.1 Detailed Description

Noop command structure.

For command details see `CF_NOOP_CC`

Definition at line 74 of file default_cf_msgstruct.h.

11.88.2 Field Documentation

11.88.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CF_NoopCmd::CommandHeader
Command header.

Definition at line 76 of file default_cf_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.89 CF_Output Struct Reference

CF engine output state.

```
#include <cf_cfdp_types.h>
```

Data Fields

- `CFE_SB_Buffer_t * msg`
Binary message to be sent to underlying transport.
- `CF_EncoderState_t encode`
Encoding state (while building message)
- `CF_Logical_PduBuffer_t tx_pdudata`
Tx PDU logical values.

11.89.1 Detailed Description

CF engine output state.

Keeps the state of the current output PDU in the CF engine

Definition at line 418 of file cf_cfdp_types.h.

11.89.2 Field Documentation

11.89.2.1 encode `CF_EncoderState_t` CF_Output::encode

Encoding state (while building message)

Definition at line 421 of file cf_cfdp_types.h.

Referenced by CF_CFDP_MsgOutGet().

11.89.2.2 msg `CFE_SB_Buffer_t*` CF_Output::msg

Binary message to be sent to underlying transport.

Definition at line 420 of file cf_cfdp_types.h.

Referenced by CF_CFDP_MsgOutGet(), and CF_CFDP_Send().

11.89.2.3 tx_pdudata `CF_Logical_PduBuffer_t` CF_Output::tx_pdudata

Tx PDU logical values.

Definition at line 422 of file cf_cfdp_types.h.

Referenced by CF_CFDP_MsgOutGet().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.90 CF_PduCmdMsg Struct Reference

PDU command encapsulation structure.

```
#include <cf_cfdp_sbintf.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `hdr`
software bus headers, not really used by CF
- [CF_CFDP_PduHeader_t](#) `ph`
Beginning of CFDP headers.

11.90.1 Detailed Description

PDU command encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "command" encapsulation (even though these are not really commands).

Note

this is only the definition of the header. In reality all messages are larger than this, up to CF_MAX_PDU_SIZE.

Definition at line 44 of file cf_cfdp_sbintf.h.

11.90.2 Field Documentation

11.90.2.1 `hdr` [CFE_MSG_CommandHeader_t](#) `CF_PduCmdMsg::hdr`
software bus headers, not really used by CF
Definition at line 46 of file cf_cfdp_sbintf.h.

11.90.2.2 `ph` [CF_CFDP_PduHeader_t](#) `CF_PduCmdMsg::ph`
Beginning of CFDP headers.

Definition at line 47 of file cf_cfdp_sbintf.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_sbintf.h](#)

11.91 CF_PduTlmMsg Struct Reference

PDU send encapsulation structure.

```
#include <cf_cfdp_sbintf.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) `hdr`
software bus headers, not really used by CF
- [CF_CFDP_PduHeader_t](#) `ph`
Beginning of CFDP headers.

11.91.1 Detailed Description

PDU send encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "telemetry" encapsulation (even though these are not really telemetry items).

Note

this is only the definition of the header. In reality all messages are larger than this, up to CF_MAX_PDU_SIZE.

Definition at line 60 of file cf_cfdp_sbintf.h.

11.91.2 Field Documentation

11.91.2.1 **hdr** [CFE_MSG_TelemetryHeader_t](#) CF_PduTlmMsg::hdr

software bus headers, not really used by CF

Definition at line 62 of file cf_cfdp_sbintf.h.

11.91.2.2 **ph** [CF_CFDP_PduHeader_t](#) CF_PduTlmMsg::ph

Beginning of CFDP headers.

Definition at line 63 of file cf_cfdp_sbintf.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_sbintf.h](#)

11.92 CF_Playback Struct Reference

CF Playback entry.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [osal_id_t dir_id](#)
- [CF_CFDP_Class_t cfdp_class](#)
- [CF_TxnFilenames_t fnames](#)
- [uint16 num_ts](#)
number of transactions
- [uint8 priority](#)
- [CF_EntityId_t dest_id](#)
- [bool busy](#)
- [bool diropen](#)
- [bool keep](#)
- [bool counted](#)

11.92.1 Detailed Description

CF Playback entry.

Keeps the state of CF playback requests

Definition at line 200 of file cf_cfdp_types.h.

11.92.2 Field Documentation

11.92.2.1 **busy** [bool](#) CF_Playback::busy

Definition at line 209 of file cf_cfdp_types.h.

Referenced by [CF_CFDP_DisableEngine\(\)](#), [CF_CFDP_PlaybackDir\(\)](#), [CF_CFDP_PlaybackDir_Initiate\(\)](#), [CF_CFDP_ProcessPlaybackDirectories\(\)](#), [CF_CFDP_ProcessPlaybackDirectory\(\)](#), and [CF_CFDP_ProcessPollingDirectories\(\)](#).

11.92.2.2 cfdp_class `CF_CFDP_Class_t` CF_Playback::cfdp_class

Definition at line 203 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.3 counted `bool` CF_Playback::counted

Definition at line 212 of file cf_cfdp_types.h.

Referenced by CF_CFDP_UpdatePollPbCounted().

11.92.2.4 dest_id `CF_EntityId_t` CF_Playback::dest_id

Definition at line 207 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.5 dir_id `osal_id_t` CF_Playback::dir_id

Definition at line 202 of file cf_cfdp_types.h.

Referenced by CF_CFDP_DisableEngine(), CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.6 diropen `bool` CF_Playback::diropen

Definition at line 210 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.7 fnames `CF_TxnFilenames_t` CF_Playback::fnames

Definition at line 204 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.8 keep `bool` CF_Playback::keep

Definition at line 211 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

11.92.2.9 num_ts `uint16` CF_Playback::num_ts

number of transactions

Definition at line 205 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_ProcessPollingDirectories(), and CF_CFDP_ResetTransaction().

11.92.2.10 priority `uint8` CF_Playback::priority

Definition at line 206 of file cf_cfdp_types.h.

Referenced by CF_CFDP_PlaybackDir_Initiate(), and CF_CFDP_ProcessPlaybackDirectory().

The documentation for this struct was generated from the following file:

- [apps/cf/fsu/src/cf_cfdp_types.h](#)

11.93 CF_PlaybackDirCmd Struct Reference

Playback directory command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_TxFile_Payload_t](#) Payload

11.93.1 Detailed Description

Playback directory command structure.

For command details see [CF_PLAYBACK_DIR_CC](#)

Definition at line 236 of file default_cf_msgstruct.h.

11.93.2 Field Documentation

11.93.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_PlaybackDirCmd::CommandHeader

Command header.

Definition at line 238 of file default_cf_msgstruct.h.

11.93.2.2 Payload [CF_TxFile_Payload_t](#) CF_PlaybackDirCmd::Payload

Definition at line 239 of file default_cf_msgstruct.h.

Referenced by CF_PlaybackDirCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default_cf_msgstruct.h

11.94 CF_Poll Struct Reference

CF Poll entry.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_Playback_t](#) pb
- [CF_Timer_t](#) interval_timer
- bool timer_set

11.94.1 Detailed Description

CF Poll entry.

Keeps the state of CF directory polling

Definition at line 220 of file cf_cfdp_types.h.

11.94.2 Field Documentation

11.94.2.1 interval_timer [CF_Timer_t](#) CF_Poll::interval_timer

Definition at line 223 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ProcessPollingDirectories().

11.94.2.2 pb [CF_Playback_t](#) CF_Poll::pb

Definition at line 222 of file cf_cfdp_types.h.

Referenced by CF_CFDP_DisableEngine(), and CF_CFDP_ProcessPollingDirectories().

11.94.2.3 timer_set bool CF_Poll::timer_set

Definition at line 224 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ProcessPollingDirectories().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf_cfdp_types.h\)](#)

11.95 CF_PollDir Struct Reference

Configuration entry for directory polling.

```
#include <default_cf_tbldefs.h>
```

Data Fields

- [uint32 interval_sec](#)
number of seconds to wait before trying a new directory
- [uint8 priority](#)
priority to use when placing transactions on the pending queue
- [CF_CFDP_Class_t cfdp_class](#)
the CFDP class to send
- [CF_EntityId_t dest_eid](#)
destination entity id
- [char src_dir \[CF_FILENAME_MAX_PATH\]](#)
path to source dir
- [char dst_dir \[CF_FILENAME_MAX_PATH\]](#)
path to destination dir
- [uint8 enabled](#)
Enabled flag.

11.95.1 Detailed Description

Configuration entry for directory polling.

Definition at line 40 of file default_cf_tbldefs.h.

11.95.2 Field Documentation

11.95.2.1 cfdp_class [CF_CFDP_Class_t](#) CF_PollDir::cfdp_class

the CFDP class to send

Definition at line 45 of file default_cf_tbldefs.h.

Referenced by CF_CFDP_ProcessPollingDirectories().

11.95.2.2 dest_eid `CF_EntityId_t` `CF_PollDir::dest_eid`
destination entity id

Definition at line 46 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

11.95.2.3 dst_dir `char` `CF_PollDir::dst_dir[CF_FILENAME_MAX_PATH]`

path to destination dir

Definition at line 49 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

11.95.2.4 enabled `uint8` `CF_PollDir::enabled`

Enabled flag.

Definition at line 51 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_ProcessPollingDirectories()`, and `CF_DoEnableDisablePolldir()`.

11.95.2.5 interval_sec `uint32` `CF_PollDir::interval_sec`

number of seconds to wait before trying a new directory

Definition at line 42 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

11.95.2.6 priority `uint8` `CF_PollDir::priority`

priority to use when placing transactions on the pending queue

Definition at line 44 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

11.95.2.7 src_dir `char` `CF_PollDir::src_dir[CF_FILENAME_MAX_PATH]`

path to source dir

Definition at line 48 of file `default_cf_tbldefs.h`.

Referenced by `CF_CFDP_IsPollingDir()`, and `CF_CFDP_ProcessPollingDirectories()`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_tbldefs.h`

11.96 CF_PurgeQueueCmd Struct Reference

PurgeQueue command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

Command header.

- `CF_UnionArgs_Payload_t Payload`

Generic command arguments.

11.96.1 Detailed Description

PurgeQueue command structure.

For command details see [CF_PURGE_QUEUE_CC](#)

Definition at line 181 of file default_cf_msgstruct.h.

11.96.2 Field Documentation

11.96.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_PurgeQueueCmd::CommandHeader

Command header.

Definition at line 183 of file default_cf_msgstruct.h.

11.96.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_PurgeQueueCmd::Payload

Generic command arguments.

Definition at line 184 of file default_cf_msgstruct.h.

Referenced by CF_PurgeQueueCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.97 CF_ResetCountersCmd Struct Reference

Reset command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
 - Command header.*
- [CF_UnionArgs_Payload_t](#) Payload
 - Generic command arguments.*

11.97.1 Detailed Description

Reset command structure.

For command details see [CF_RESET_CC](#)

Definition at line 104 of file default_cf_msgstruct.h.

11.97.2 Field Documentation

11.97.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_ResetCountersCmd::CommandHeader

Command header.

Definition at line 106 of file default_cf_msgstruct.h.

11.97.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_ResetCountersCmd::Payload
Generic command arguments.

Definition at line 107 of file default_cf_msgstruct.h.

Referenced by CF_ResetCountersCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.98 CF_ResumeCmd Struct Reference

Resume command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CF_Transaction_Payload_t Payload](#)

11.98.1 Detailed Description

Resume command structure.

For command details see [CF_RESUME_CC](#)

Definition at line 258 of file default_cf_msgstruct.h.

11.98.2 Field Documentation

11.98.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_ResumeCmd::CommandHeader
Command header.

Definition at line 260 of file default_cf_msgstruct.h.

11.98.2.2 Payload [CF_Transaction_Payload_t](#) CF_ResumeCmd::Payload

Definition at line 261 of file default_cf_msgstruct.h.

Referenced by CF_ResumeCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.99 CF_RxS2_Data Struct Reference

Data specific to a class 2 receive file transaction.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [uint32 eof_crc](#)
- [uint32 eof_size](#)
- [uint32 rx_crc_calc_bytes](#)
- [CF_CFDP_FinDeliveryCode_t dc](#)
- [CF_CFDP_FinFileStatus_t fs](#)
- [uint8 eof_cc](#)
remember the cc in the received EOF PDU to echo in eof-ack
- [uint8 acknak_count](#)

11.99.1 Detailed Description

Data specific to a class 2 receive file transaction.
Definition at line 250 of file cf_cfdp_types.h.

11.99.2 Field Documentation

11.99.2.1 acknak_count `uint8 CF_RxS2_Data::acknak_count`

Definition at line 258 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_R_Tick().

11.99.2.2 dc `CF_CFDP_FinDeliveryCode_t CF_RxS2_Data::dc`

Definition at line 255 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_SubstateSendFin(), and CF_CFDP_ReceiveMessage().

11.99.2.3 eof_cc `uint8 CF_RxS2_Data::eof_cc`

remember the cc in the received EOF PDU to echo in eof-ack
Definition at line 257 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateRecvEof(), and CF_CFDP_R_Tick().

11.99.2.4 eof_crc `uint32 CF_RxS2_Data::eof_crc`

Definition at line 252 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_R2_SubstateRecvEof().

11.99.2.5 eof_size `uint32 CF_RxS2_Data::eof_size`

Definition at line 253 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_RecvMd(), and CF_CFDP_R2_SubstateRecvEof().

11.99.2.6 fs `CF_CFDP_FinFileStatus_t CF_RxS2_Data::fs`

Definition at line 256 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_SubstateSendFin(), and CF_CFDP_ReceiveMessage().

11.99.2.7 rx_crc_calc_bytes `uint32 CF_RxS2_Data::rx_crc_calc_bytes`

Definition at line 254 of file cf_cfdp_types.h.
Referenced by CF_CFDP_R2_CalcCrcChunk().
The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_cfdp_types.h](#)

11.100 CF_RxState_Data Struct Reference

Data specific to a receive file transaction.
`#include <cf_cfdp_types.h>`

Data Fields

- [CF_RxSubState_t](#) sub_state
- [uint32](#) cached_pos
- [CF_RxS2_Data_t](#) r2

11.100.1 Detailed Description

Data specific to a receive file transaction.

Definition at line 264 of file cf_cfdp_types.h.

11.100.2 Field Documentation**11.100.2.1 cached_pos [uint32](#) CF_RxState_Data::cached_pos**

Definition at line 267 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_RecvMd(), and CF_CFDP_R_ProcessFd().

11.100.2.2 r2 [CF_RxS2_Data_t](#) CF_RxState_Data::r2

Definition at line 269 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Tick(), and CF_CFDP_ReceiveMessage().

11.100.2.3 sub_state [CF_RxSubState_t](#) CF_RxState_Data::sub_state

Definition at line 266 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Cancel(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), and CF_CFDP_R_Tick().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.101 CF_SendHkCmd Struct Reference

Send Housekeeping Command.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.101.1 Detailed Description

Send Housekeeping Command.

Internal notification from SCH with no payload

Definition at line 291 of file default_cf_msgstruct.h.

11.101.2 Field Documentation

11.101.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CF_SendHkCmd::CommandHeader`
Command header.

Definition at line 293 of file default_cf_msgstruct.h.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

11.102 CF_SetParam_Payload Struct Reference

Set parameter command structure.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint32 value`
Parameter value to set.
- `uint8 key`
Parameter key, see [CF_Set_ValueID_t](#).
- `uint8 chan_num`
Channel number.
- `uint8 spare [2]`
Alignment spare, uint32 multiple.

11.102.1 Detailed Description

Set parameter command structure.

For command details see [CF_SET_PARAM_CC](#)

Definition at line 228 of file default_cf_msgdefs.h.

11.102.2 Field Documentation

11.102.2.1 chan_num `uint8` `CF_SetParam_Payload::chan_num`

Channel number.

Definition at line 232 of file default_cf_msgdefs.h.

Referenced by `CF_SetParamCmd()`.

11.102.2.2 key `uint8` `CF_SetParam_Payload::key`

Parameter key, see [CF_Set_ValueID_t](#).

Definition at line 231 of file default_cf_msgdefs.h.

Referenced by `CF_SetParamCmd()`.

11.102.2.3 spare `uint8` `CF_SetParam_Payload::spare[2]`

Alignment spare, uint32 multiple.

Definition at line 233 of file default_cf_msgdefs.h.

11.102.2.4 value uint32 CF_SetParam_Payload::value

Parameter value to set.

Definition at line 230 of file default_cf_msgdefs.h.

Referenced by CF_SetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgdefs.h](#)

11.103 CF_SetParamCmd Struct Reference

Set parameter command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CF_SetParam_Payload_t Payload](#)

11.103.1 Detailed Description

Set parameter command structure.

For command details see [CF_SET_PARAM_CC](#)

Definition at line 203 of file default_cf_msgstruct.h.

11.103.2 Field Documentation

11.103.2.1 CommandHeader [CFE_MSG_CommandHeader_t CF_SetParamCmd::CommandHeader](#)

Command header.

Definition at line 205 of file default_cf_msgstruct.h.

11.103.2.2 Payload [CF_SetParam_Payload_t CF_SetParamCmd::Payload](#)

Definition at line 206 of file default_cf_msgstruct.h.

Referenced by CF_SetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.104 CF_StateData Union Reference

Summary of all possible transaction state information (tx and rx)

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_TxState_Data_t send](#)
applies to only send file transactions
- [CF_RxState_Data_t receive](#)
applies to only receive file transactions

11.104.1 Detailed Description

Summary of all possible transaction state information (tx and rx)
Definition at line 325 of file cf_cfdp_types.h.

11.104.2 Field Documentation

11.104.2.1 receive `CF_RxState_Data_t` CF_StateData::receive

applies to only receive file transactions

Definition at line 328 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Cancel(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_Tick(), and CF_CFDP_ReceiveMessage().

11.104.2.2 send `CF_TxState_Data_t` CF_StateData::send

applies to only send file transactions

Definition at line 327 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S2_Fin(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_Cancel(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().

The documentation for this union was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.105 CF_StateFlags Union Reference

Summary of all possible transaction flags (tx and rx)

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_Flags_Common_t com](#)
applies to all transactions
- [CF_Flags_Rx_t rx](#)
applies to only receive file transactions
- [CF_Flags_Tx_t tx](#)
applies to only send file transactions

11.105.1 Detailed Description

Summary of all possible transaction flags (tx and rx)
Definition at line 315 of file cf_cfdp_types.h.

11.105.2 Field Documentation

11.105.2.1 com CF_Flags_Common_t CF_StateFlags::com

applies to all transactions

Definition at line 317 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_CancelTransaction(), CF_CFDP_CycleTxFirstActive(), CF_CFDP_DoTick(), CF_CFDP_GetClass(), CF_CFDP_IsSender(), CF_CFDP_MsgOutGet(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_ResetTransaction(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_SendEof(), CF_CFDP_S_Tick(), CF_DequeueTransaction(), CF_DoSuspRes_Txn(), CF_FreeTransaction(), CF_InsertSortPrio(), and CF_MoveTransaction().

11.105.2.2 rx CF_Flags_Rx_t CF_StateFlags::rx

applies to only receive file transactions

Definition at line 318 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_Cancel(), CF_CFDP_R_Init(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), and CF_CFDP_RecvIdle().

11.105.2.3 tx CF_Flags_Tx_t CF_StateFlags::tx

applies to only send file transactions

Definition at line 319 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ResetTransaction(), CF_CFDP_S2_Nak(), CF_CFDP_S_CheckAndRespondNak(), and CF_CFDP_TxFile().

The documentation for this union was generated from the following file:

- apps/cf/fsw/src/[cf_cfdp_types.h](#)

11.106 CF_SuspendCmd Struct Reference

Suspend command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CF_Transaction_Payload_t Payload](#)

11.106.1 Detailed Description

Suspend command structure.

For command details see [CF_SUSPEND_CC](#)

Definition at line 247 of file default_cf_msgstruct.h.

11.106.2 Field Documentation

11.106.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_SuspendCmd::CommandHeader

Command header.

Definition at line 249 of file default_cf_msgstruct.h.

11.106.2.2 Payload [CF_Transaction_Payload_t](#) CF_SuspendCmd::Payload

Definition at line 250 of file default_cf_msgstruct.h.

Referenced by CF_SuspendCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.107 CF_ThawCmd Struct Reference

Thaw command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CF_UnionArgs_Payload_t](#) Payload
Generic command arguments.

11.107.1 Detailed Description

Thaw command structure.

For command details see [CF_THAW_CC](#)

Definition at line 126 of file default_cf_msgstruct.h.

11.107.2 Field Documentation

11.107.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_ThawCmd::CommandHeader

Command header.

Definition at line 128 of file default_cf_msgstruct.h.

11.107.2.2 Payload [CF_UnionArgs_Payload_t](#) CF_ThawCmd::Payload

Generic command arguments.

Definition at line 129 of file default_cf_msgstruct.h.

Referenced by CF_ThawCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.108 CF_Timer Struct Reference

Basic CF timer object.

```
#include <cf_timer.h>
```

Data Fields

- [CF_Timer_Ticks_t](#) tick
expires when reaches 0

11.108.1 Detailed Description

Basic CF timer object.

Definition at line 48 of file cf_timer.h.

11.108.2 Field Documentation

11.108.2.1 tick [CF_Timer_Ticks_t](#) CF_Timer::tick

expires when reaches 0

Definition at line 50 of file cf_timer.h.

Referenced by CF_Timer_Expired(), CF_Timer_InitRelSec(), and CF_Timer_Tick().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_timer.h](#)

11.109 CF_Transaction Struct Reference

Transaction state object.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_TxnState_t](#) state
 - each engine is commanded to do something, which is the overall state*
- [CF_History_t](#) * history
 - weird, holds active filenames and possibly other info*
- [CF_ChunkWrapper_t](#) * chunks
 - for gap tracking, only used on class 2*
- [CF_Timer_t](#) inactivity_timer
 - set to the overall inactivity timer of a remote*
- [CF_Timer_t](#) ack_timer
 - called ack_timer, but is also nak_timer*
- uint32 fsize
 - lseek() should be 64-bit on 64-bit system, but osal limits to 32-bit*
- uint32 foofs
 - offset into file for next read*
- [osal_id_t](#) fd
- [CF_Crc_t](#) crc
- uint8 keep
- uint8 chan_num
 - if ever more than one engine, this may need to change to pointer*
- uint8 priority
- [CF_CListNode_t](#) cl_node
- [CF_Playback_t](#) * pb
 - NULL if transaction does not belong to a playback.*
- [CF_StateData_t](#) state_data
- [CF_StateFlags_t](#) flags
 - State flags.*

11.109.1 Detailed Description

Transaction state object.

This keeps the state of CF file transactions

Definition at line 336 of file cf_cfdp_types.h.

11.109.2 Field Documentation

11.109.2.1 ack_timer `CF_Timer_t` `CF_Transaction::ack_timer`

called ack_timer, but is also nak_timer

Definition at line 343 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_ArmAckTimer()`, `CF_CFDP_R_Tick()`, and `CF_CFDP_S_Tick()`.

11.109.2.2 chan_num `uint8` `CF_Transaction::chan_num`

if ever more than one engine, this may need to change to pointer

Definition at line 352 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_ArmAckTimer()`, `CF_CFDP_ArmInactTimer()`, `CF_CFDP_CycleTxFirstActive()`, `CF_CFDP_HandleNotKeepFile()`, `CF_CFDP_InitEngine()`, `CF_CFDP_InitTxnTxFile()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_R2_CalcCrcChunk()`, `CF_CFDP_R2_Complete()`, `CF_CFDP_R2_Recv_fin_ack()`, `CF_CFDP_R2_RecvMd()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_R_Init()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendInactivityEvent()`, `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_R_SubstateSendNak()`, `CF_CFDP_R_Tick()`, `CF_CFDP_RecvDrop()`, `CF_CFDP_RecvFd()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_RecvMd()`, `CF_CFDP_ResetTransaction()`, `CF_CFDP_S2_Nak()`, `CF_CFDP_S2_WaitForEofAck()`, `CF_CFDP_S_DispatchRecv()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_S_SubstateSendMetadata()`, `CF_CFDP_S_Tick()`, `CF_CFDP_SendAck()`, `CF_CFDP_SendEof()`, `CF_CFDP_SendEotPkt()`, `CF_CFDP_SendFd()`, `CF_CFDP_SendFin()`, `CF_CFDP_SendMd()`, `CF_CFDP_SendNak()`, `CF_DequeueTransaction()`, `CF_FreeTransaction()`, `CF_InsertSortPrio()`, and `CF_MoveTransaction()`.

11.109.2.3 chunks `CF_ChunkWrapper_t*` `CF_Transaction::chunks`

for gap tracking, only used on class 2

Definition at line 341 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_R2_Complete()`, `CF_CFDP_R2_SubstateRecvFileData()`, `CF_CFDP_R_SubstateSendNak()`, `CF_CFDP_RecvIdle()`, `CF_CFDP_ResetTransaction()`, `CF_CFDP_S2_Nak()`, `CF_CFDP_S_CheckAndRespondNak()`, and `CF_CFDP_TxFile_Initiate()`.

11.109.2.4 cl_node `CF_CListNode_t` `CF_Transaction::cl_node`

Definition at line 355 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_ReceiveMessage()`, `CF_DequeueTransaction()`, `CF_FindUnusedTransaction()`, `CF_FreeTransaction()`, `CF_InsertSortPrio()`, and `CF_MoveTransaction()`.

11.109.2.5 crc `CF_Crc_t` `CF_Transaction::crc`

Definition at line 349 of file cf_cfdp_types.h.

Referenced by `CF_CFDP_R1_SubstateRecvFileData()`, `CF_CFDP_R2_CalcCrcChunk()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_S_SendEof()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_SendEof()`, and `CF_CFDP_SendEotPkt()`.

11.109.2.6 fd `osal_id_t` `CF_Transaction::fd`

Definition at line 347 of file cf_cfdp_types.h.

Referenced by CF_CFDP_CloseFiles(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_ResetTransaction(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), and CF_FreeTransaction().

11.109.2.7 flags `CF_StateFlags_t` CF_Transaction::flags

State flags.

Note

The flags here look a little strange, because there are different flags for TX and RX. Both types share the same type of flag, though. Since RX flags plus the global flags is over one byte, storing them this way allows 2 bytes to cover all possible flags. Please ignore the duplicate declarations of the "all" flags.

Definition at line 369 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_CancelTransaction(), CF_CFDP_CycleTxFirstActive(), CF_CFDP_DoTick(), CF_CFDP_GetClass(), CF_CFDP_IsSender(), CF_CFDP_MsgOutGet(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Cancel(), CF_CFDP_R_Init(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_ResetTransaction(), CF_CFDP_S2_Nak(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_CheckAndRespondNak(), CF_CFDP_S_SendEof(), CF_CFDP_S_Tick(), CF_CFDP_TxFile(), CF_DequeueTransaction(), CF_DoSuspRes_Txn(), CF_FreeTransaction(), CF_InsertSortPrio(), and CF_MoveTransaction().

11.109.2.8 foofs `uint32` CF_Transaction::foofs

offset into file for next read

Definition at line 346 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S_SubstateSendFileData().

11.109.2.9 fsize `uint32` CF_Transaction::fsize

Iseek() should be 64-bit on 64-bit system, but osal limits to 32-bit

Definition at line 345 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_RecvMd(), CF_CFDP_S2_Nak(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), and CF_CFDP_SendMd().

11.109.2.10 history `CF_History_t*` CF_Transaction::history

weird, holds active filenames and possibly other info

Definition at line 340 of file cf_cfdp_types.h.

Referenced by CF_CFDP_HandleNotKeepFile(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_ResetTransaction(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_SendEof(), CF_CFDP_SendEotPkt(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_SetTxnStatus(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), CF_FindTransactionBySequenceNumber_Impl(), CF_FindUnusedTransaction(), and CF_Traverse_WriteTxnQueueEntryToFile().

11.109.2.11 inactivity_timer [CF_Timer_t](#) CF_Transaction::inactivity_timer

set to the overall inactivity timer of a remote

Definition at line 342 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ArmInactTimer(), CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

11.109.2.12 keep [uint8](#) CF_Transaction::keep

Definition at line 351 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitTxnTxFile(), CF_CFDP_R1_SubstateRecvEof(), CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_ResetTransaction().

11.109.2.13 pb [CF_Playback_t*](#) CF_Transaction::pb

NULL if transaction does not belong to a playback.

Definition at line 357 of file cf_cfdp_types.h.

Referenced by CF_CFDP_ProcessPlaybackDirectory(), and CF_CFDP_ResetTransaction().

11.109.2.14 priority [uint8](#) CF_Transaction::priority

Definition at line 353 of file cf_cfdp_types.h.

Referenced by CF_CFDP_InitTxnTxFile(), CF_InsertSortPrio(), and CF_PrioSearch().

11.109.2.15 state [CF_TxnState_t](#) CF_Transaction::state

each engine is commanded to do something, which is the overall state

Definition at line 338 of file cf_cfdp_types.h.

Referenced by CF_CFDP_GetClass(), CF_CFDP_InitTxnTxFile(), CF_CFDP_IsSender(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_Cancel(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvIdle(), CF_CFDP_RxStateDispatch(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForeofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_SendEotPkt(), CF_CFDP_SendMd(), CF_CFDP_TxFile(), CF_CFDP_TxStateDispatch(), CF_FreeTransaction(), and CF_InsertSortPrio().

11.109.2.16 state_data [CF_StateData_t](#) CF_Transaction::state_data

Definition at line 359 of file cf_cfdp_types.h.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_Cancel(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_S2_Fin(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForeofAck(), CF_CFDP_S_Cancel(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_cfdp_types.h](#)

11.110 CF_Transaction_Payload Struct Reference

Transaction command structure.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- **CF_TransactionSeq_t ts**
Transaction sequence number.
- **CF_EntityId_t eid**
Entity id.
- **uint8 chan**
Channel number: 254=use ts, 255=all channels, else channel.
- **uint8 spare [3]**
Alignment spare for 32-bit multiple.

11.110.1 Detailed Description

Transaction command structure.

For command details see [CF_SUSPEND_CC](#), [CF_RESUME_CC](#), [CF_CANCEL_CC](#), [CF_ABANDON_CC](#)

Definition at line 272 of file default_cf_msgdefs.h.

11.110.2 Field Documentation

11.110.2.1 chan `uint8 CF_Transaction_Payload::chan`

Channel number: 254=use ts, 255=all channels, else channel.

Definition at line 276 of file default_cf_msgdefs.h.

Referenced by [CF_TsnChanAction\(\)](#).

11.110.2.2 eid `CF_EntityId_t CF_Transaction_Payload::eid`

Entity id.

Definition at line 275 of file default_cf_msgdefs.h.

Referenced by [CF_TsnChanAction\(\)](#).

11.110.2.3 spare `uint8 CF_Transaction_Payload::spare[3]`

Alignment spare for 32-bit multiple.

Definition at line 277 of file default_cf_msgdefs.h.

11.110.2.4 ts `CF_TransactionSeq_t CF_Transaction_Payload::ts`

Transaction sequence number.

Definition at line 274 of file default_cf_msgdefs.h.

Referenced by [CF_TsnChanAction\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgdefs.h](#)

11.111 CF_Traverse_PriorityArg Struct Reference

Argument structure for use with [CF_CList_Traverse_R\(\)](#)

```
#include <cf_utils.h>
```

Data Fields

- [CF_Transaction_t * txn](#)
OUT: holds value of transaction with which to call CF_CList_InsertAfter on.
- [uint8 priority](#)
seeking this priority

11.111.1 Detailed Description

Argument structure for use with [CF_CList_Traverse_R\(\)](#)

This is for searching for transactions of a specific priority

Definition at line 107 of file cf_utils.h.

11.111.2 Field Documentation

11.111.2.1 priority [uint8 CF_Traverse_PriorityArg::priority](#)

seeking this priority

Definition at line 110 of file cf_utils.h.

Referenced by [CF_PrioSearch\(\)](#).

11.111.2.2 txn [CF_Transaction_t* CF_Traverse_PriorityArg::txn](#)

OUT: holds value of transaction with which to call CF_CList_InsertAfter on.

Definition at line 109 of file cf_utils.h.

Referenced by [CF_InsertSortPrio\(\)](#), and [CF_PrioSearch\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf_utils.h](#)

11.112 CF_Traverse_TransSeqArg Struct Reference

Argument structure for use with [CList_Traverse\(\)](#)

```
#include <cf_utils.h>
```

Data Fields

- [CF_TransactionSeq_t transaction_sequence_number](#)
- [CF_EntityId_t src_eid](#)
- [CF_Transaction_t * txn](#)
output transaction pointer

11.112.1 Detailed Description

Argument structure for use with [CList_Traverse\(\)](#)

This identifies a specific transaction sequence number and entity ID The transaction pointer is set by the implementation

Definition at line 39 of file cf_utils.h.

11.112.2 Field Documentation

11.112.2.1 src_eid CF_EntityId_t CF_Traverse_TransSeqArg::src_eid

Definition at line 42 of file cf_utils.h.

Referenced by CF_FindTransactionBySequenceNumber_Impl().

11.112.2.2 transaction_sequence_number CF_TransactionSeq_t CF_Traverse_TransSeqArg::transaction_sequence_number

Definition at line 41 of file cf_utils.h.

Referenced by CF_FindTransactionBySequenceNumber_Impl().

11.112.2.3 txn CF_Transaction_t* CF_Traverse_TransSeqArg::txn

output transaction pointer

Definition at line 43 of file cf_utils.h.

Referenced by CF_FindTransactionBySequenceNumber(), and CF_FindTransactionBySequenceNumber_Impl().

The documentation for this struct was generated from the following file:

- [apps/cf/fsu/src/cf_utils.h](#)

11.113 CF_Traverse_WriteHistoryFileArg Struct Reference

Argument structure for use with [CF_Traverse_WriteHistoryQueueEntryToFile\(\)](#)

```
#include <cf_utils.h>
```

Data Fields

- [osal_id_t fd](#)
- [CF_Direction_t filter_dir](#)
- bool [error](#)

Will be set to true if any write failed.
- uint32 [counter](#)

Total number of entries written.

11.113.1 Detailed Description

Argument structure for use with [CF_Traverse_WriteHistoryQueueEntryToFile\(\)](#)

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing history, the list contains all entries, and may need additional filtering for direction (TX/RX) depending on what information the user has requested.

Definition at line 55 of file cf_utils.h.

11.113.2 Field Documentation

11.113.2.1 counter uint32 CF_Traverse_WriteHistoryFileArg::counter

Total number of entries written.

Definition at line 61 of file cf_utils.h.

Referenced by CF_Traverse_WriteHistoryQueueEntryToFile(), and CF_WriteHistoryQueueDataToFile().

11.113.2.2 error bool CF_Traverse_WriteHistoryFileArg::error

Will be set to true if any write failed.

Definition at line 60 of file cf_utils.h.

Referenced by CF_Traverse_WriteHistoryQueueEntryToFile(), and CF_WriteHistoryQueueDataToFile().

11.113.2.3 fd osal_id_t CF_Traverse_WriteHistoryFileArg::fd

Definition at line 57 of file cf_utils.h.

Referenced by CF_Traverse_WriteHistoryQueueEntryToFile(), and CF_WriteHistoryQueueDataToFile().

11.113.2.4 filter_dir CF_Direction_t CF_Traverse_WriteHistoryFileArg::filter_dir

Definition at line 58 of file cf_utils.h.

Referenced by CF_Traverse_WriteHistoryQueueEntryToFile(), and CF_WriteHistoryQueueDataToFile().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_utils.h](#)

11.114 CF_Traverse_WriteTxnFileArg Struct Reference

Argument structure for use with [CF_Traverse_WriteTxnQueueEntryToFile\(\)](#)

```
#include <cf_utils.h>
```

Data Fields

- **osal_id_t fd**
- **bool error**

Will be set to true if any write failed.

- **uint32 counter**

Total number of entries written.

11.114.1 Detailed Description

Argument structure for use with [CF_Traverse_WriteTxnQueueEntryToFile\(\)](#)

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing transactions, the entire list is written to the file. No additional filtering is necessary, because the queues themselves are limited in what they contain (therefore "pre-filtered" to some degree).

Definition at line 74 of file cf_utils.h.

11.114.2 Field Documentation

11.114.2.1 counter uint32 CF_Traverse_WriteTxnFileArg::counter

Total number of entries written.

Definition at line 79 of file cf_utils.h.

Referenced by CF_Traverse_WriteTxnQueueEntryToFile(), and CF_WriteTxnQueueDataToFile().

11.114.2.2 error bool CF_Traverse_WriteTxnFileArg::error

Will be set to true if any write failed.

Definition at line 78 of file cf_utils.h.

Referenced by CF_Traverse_WriteTxnQueueEntryToFile(), and CF_WriteTxnQueueDataToFile().

11.114.2.3 fd osal_id_t CF_Traverse_WriteTxnFileArg::fd

Definition at line 76 of file cf_utils.h.

Referenced by CF_Traverse_WriteTxnQueueEntryToFile(), and CF_WriteTxnQueueDataToFile().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_utils.h](#)

11.115 CF_TraverseAll_Arg Struct Reference

Argument structure for use with [CF_TraverseAllTransactions\(\)](#)

```
#include <cf_utils.h>
```

Data Fields

- [CF_TraverseAllTransactions_fn_t fn](#)
internal callback to use for each CList_Traverse
- void * [context](#)
opaque object to pass to internal callback
- int32 [counter](#)
Running tally of all nodes traversed from all lists.

11.115.1 Detailed Description

Argument structure for use with [CF_TraverseAllTransactions\(\)](#)

This basically allows for running a CF_Traverse on several lists at once

Definition at line 95 of file cf_utils.h.

11.115.2 Field Documentation

11.115.2.1 context void* CF_TraverseAll_Arg::context

opaque object to pass to internal callback

Definition at line 98 of file cf_utils.h.

Referenced by CF_TraverseAllTransactions_Impl().

11.115.2.2 counter int32 CF_TraverseAll_Arg::counter

Running tally of all nodes traversed from all lists.

Definition at line 99 of file cf_utils.h.

Referenced by CF_TraverseAllTransactions(), and CF_TraverseAllTransactions_Impl().

11.115.2.3 fn CF_TraverseAllTransactions_fn_t CF_TraverseAll_Arg::fn

internal callback to use for each CList_Traverse

Definition at line 97 of file cf_utils.h.

Referenced by CF_TraverseAllTransactions_Impl().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf_utils.h](#)

11.116 CF_TxFile_Payload Struct Reference

Transmit file command structure.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- `uint8 cfcp_class`
CFDP class: 0=class 1, 1=class 2.
- `uint8 keep`
Keep file flag: 1=keep, else delete.
- `uint8 chan_num`
Channel number.
- `uint8 priority`
Priority: 0=highest priority.
- `CF_EntityId_t dest_id`
Destination entity id.
- `char src_filename [CF_FILENAME_MAX_LEN]`
Source file/directory name.
- `char dst_filename [CF_FILENAME_MAX_LEN]`
Destination file/directory name.

11.116.1 Detailed Description

Transmit file command structure.

For command details see [CF_TX_FILE_CC](#)

Definition at line 241 of file default_cf_msgdefs.h.

11.116.2 Field Documentation

11.116.2.1 `cfcp_class uint8 CF_TxFile_Payload::cfcp_class`

CFDP class: 0=class 1, 1=class 2.

Definition at line 243 of file default_cf_msgdefs.h.

Referenced by [CF_PlaybackDirCmd\(\)](#), and [CF_TxFileCmd\(\)](#).

11.116.2.2 `chan_num uint8 CF_TxFile_Payload::chan_num`

Channel number.

Definition at line 245 of file default_cf_msgdefs.h.

Referenced by [CF_PlaybackDirCmd\(\)](#), and [CF_TxFileCmd\(\)](#).

11.116.2.3 `dest_id CF_EntityId_t CF_TxFile_Payload::dest_id`

Destination entity id.

Definition at line 247 of file default_cf_msgdefs.h.

Referenced by [CF_PlaybackDirCmd\(\)](#), and [CF_TxFileCmd\(\)](#).

11.116.2.4 `dst_filename char CF_TxFile_Payload::dst_filename [CF_FILENAME_MAX_LEN]`

Destination file/directory name.

Definition at line 249 of file default_cf_msgdefs.h.

Referenced by [CF_PlaybackDirCmd\(\)](#), and [CF_TxFileCmd\(\)](#).

11.116.2.5 keep `uint8` CF_TxFile_Payload::keep

Keep file flag: 1=keep, else delete.

Definition at line 244 of file default_cf_msgdefs.h.

Referenced by CF_PlaybackDirCmd(), and CF_TxFileCmd().

11.116.2.6 priority `uint8` CF_TxFile_Payload::priority

Priority: 0=highest priority.

Definition at line 246 of file default_cf_msgdefs.h.

Referenced by CF_PlaybackDirCmd(), and CF_TxFileCmd().

11.116.2.7 src_filename `char` CF_TxFile_Payload::src_filename[`CF_FILENAME_MAX_LEN`]

Source file/directory name.

Definition at line 248 of file default_cf_msgdefs.h.

Referenced by CF_PlaybackDirCmd(), and CF_TxFileCmd().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.117 CF_TxFileCmd Struct Reference

Transmit file command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader
Command header.
- `CF_TxFile_Payload_t` Payload

11.117.1 Detailed Description

Transmit file command structure.

For command details see `CF_TX_FILE_CC`

Definition at line 214 of file default_cf_msgstruct.h.

11.117.2 Field Documentation

11.117.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CF_TxFileCmd::CommandHeader

Command header.

Definition at line 216 of file default_cf_msgstruct.h.

11.117.2.2 Payload `CF_TxFile_Payload_t` CF_TxFileCmd::Payload

Definition at line 217 of file default_cf_msgstruct.h.

Referenced by CF_TxFileCmd().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

11.118 CF_TxnFilenames Struct Reference

Cache of source and destination filename.

```
#include <default_cf_extern_typedefs.h>
```

Data Fields

- char [src_filename](#) [[CF_FILENAME_MAX_LEN](#)]
- char [dst_filename](#) [[CF_FILENAME_MAX_LEN](#)]

11.118.1 Detailed Description

Cache of source and destination filename.

This pairs a source and destination file name together to be retained for future reference in the transaction/history
Definition at line 90 of file default_cf_extern_typedefs.h.

11.118.2 Field Documentation

11.118.2.1 dst_filename char CF_TxnFilenames::dst_filename [[CF_FILENAME_MAX_LEN](#)]

Definition at line 93 of file default_cf_extern_typedefs.h.

Referenced by [CF_CFDP_HandleNotKeepFile\(\)](#), [CF_CFDP_PlaybackDir_Initiate\(\)](#), [CF_CFDP_ProcessPlaybackDirectory\(\)](#), [CF_CFDP_R2_RecvMd\(\)](#), [CF_CFDP_R_Init\(\)](#), [CF_CFDP_RecvMd\(\)](#), [CF_CFDP_SendMd\(\)](#), [CF_CFDP_TxFile\(\)](#), [CF_CFDP_TxFile_Initiate\(\)](#), and [CF_WriteHistoryEntryToFile\(\)](#).

11.118.2.2 src_filename char CF_TxnFilenames::src_filename [[CF_FILENAME_MAX_LEN](#)]

Definition at line 92 of file default_cf_extern_typedefs.h.

Referenced by [CF_CFDP_HandleNotKeepFile\(\)](#), [CF_CFDP_PlaybackDir_Initiate\(\)](#), [CF_CFDP_ProcessPlaybackDirectory\(\)](#), [CF_CFDP_RecvMd\(\)](#), [CF_CFDP_S_SubstateSendMetadata\(\)](#), [CF_CFDP_SendMd\(\)](#), [CF_CFDP_TxFile\(\)](#), [CF_CFDP_TxFile_Initiate\(\)](#), and [CF_WriteHistoryEntryToFile\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_extern_typedefs.h](#)

11.119 CF_TxS2_Data Struct Reference

Data specific to a class 2 send file transaction.

```
#include <cf_cfdp_types.h>
```

Data Fields

- uint8 fin_cc
 - remember the cc in the received FIN PDU to echo in eof-fin*
- uint8 acknak_count

11.119.1 Detailed Description

Data specific to a class 2 send file transaction.

Definition at line 230 of file cf_cfdp_types.h.

11.119.2 Field Documentation

11.119.2.1 acknak_count uint8 CF_TxS2_Data::acknak_count

Definition at line 233 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S_Tick().

11.119.2.2 fin_cc uint8 CF_TxS2_Data::fin_cc

remember the cc in the received FIN PDU to echo in eof-fin

Definition at line 232 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S2_Fin(), and CF_CFDP_S_SubstateSendFinAck().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.120 CF_TxState_Data Struct Reference

Data specific to a send file transaction.

```
#include <cf_cfdp_types.h>
```

Data Fields

- [CF_TxSubState_t sub_state](#)
- [uint32 cached_pos](#)
- [CF_TxS2_Data_t s2](#)

11.120.1 Detailed Description

Data specific to a send file transaction.

Definition at line 239 of file cf_cfdp_types.h.

11.120.2 Field Documentation

11.120.2.1 cached_pos uint32 CF_TxState_Data::cached_pos

Definition at line 242 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S_SendFileData().

11.120.2.2 s2 CF_TxS2_Data_t CF_TxState_Data::s2

Definition at line 244 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S2_Fin(), CF_CFDP_S_SubstateSendFinAck(), and CF_CFDP_S_Tick().

11.120.2.3 sub_state CF_TxSubState_t CF_TxState_Data::sub_state

Definition at line 241 of file cf_cfdp_types.h.

Referenced by CF_CFDP_S2_Fin(), CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_Cancel(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf_cfdp_types.h](#)

11.121 CF_UnionArgs_Payload Union Reference

Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- **uint32 dword**
Generic uint32 argument.
- **uint16 hword [2]**
Generic uint16 array of arguments.
- **uint8 byte [4]**
Generic uint8 array of arguments.

11.121.1 Detailed Description

Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

Definition at line 151 of file default_cf_msgdefs.h.

11.121.2 Field Documentation

11.121.2.1 **byte** `uint8` CF_UnionArgs_Payload::byte[4]

Generic uint8 array of arguments.

Definition at line 155 of file default_cf_msgdefs.h.

Referenced by CF_DoChanAction(), CF_DoEnableDisablePolldir(), CF_DoPurgeQueue(), and CF_ResetCountersCmd().

11.121.2.2 **dword** `uint32` CF_UnionArgs_Payload::dword

Generic uint32 argument.

Definition at line 153 of file default_cf_msgdefs.h.

11.121.2.3 **hword** `uint16` CF_UnionArgs_Payload::hword[2]

Generic uint16 array of arguments.

Definition at line 154 of file default_cf_msgdefs.h.

The documentation for this union was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.122 CF_WakeupCmd Struct Reference

Wake Up Command.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t CommandHeader**
Command header.

11.122.1 Detailed Description

Wake Up Command.

Internal notification from SCH with no payload

Definition at line 301 of file default_cf_msgstruct.h.

11.122.2 Field Documentation

11.122.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CF_WakeupCmd::CommandHeader

Command header.

Definition at line 303 of file default_cf_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default_cf_msgstruct.h](#)

11.123 CF_WriteQueue_Payload Struct Reference

Write Queue command structure.

```
#include <default_cf_msgdefs.h>
```

Data Fields

- [uint8 type](#)

Transaction direction: all=0, up=1, down=2.

- [uint8 chan](#)

Channel number.

- [uint8 queue](#)

Queue type: 0=pending, 1=active, 2=history, 3=all.

- [uint8 spare](#)

Alignment spare, puts filename on 32-bit boundary.

- [char filename \[CF_FILENAME_MAX_LEN\]](#)

Filename written to.

11.123.1 Detailed Description

Write Queue command structure.

For command details see [CF_WRITE_QUEUE_CC](#)

Definition at line 257 of file default_cf_msgdefs.h.

11.123.2 Field Documentation

11.123.2.1 chan [uint8](#) CF_WriteQueue_Payload::chan

Channel number.

Definition at line 260 of file default_cf_msgdefs.h.

Referenced by CF_WriteQueueCmd().

11.123.2.2 filename `char CF_WriteQueue_Payload::filename[CF_FILENAME_MAX_LEN]`
Filename written to.
Definition at line 264 of file default_cf_msgdefs.h.
Referenced by CF_WriteQueueCmd().

11.123.2.3 queue `uint8 CF_WriteQueue_Payload::queue`
Queue type: 0=pending, 1=active, 2=history, 3=all.
Definition at line 261 of file default_cf_msgdefs.h.
Referenced by CF_WriteQueueCmd().

11.123.2.4 spare `uint8 CF_WriteQueue_Payload::spare`
Alignment spare, puts filename on 32-bit boundary.
Definition at line 262 of file default_cf_msgdefs.h.

11.123.2.5 type `uint8 CF_WriteQueue_Payload::type`
Transaction direction: all=0, up=1, down=2.
Definition at line 259 of file default_cf_msgdefs.h.
Referenced by CF_WriteQueueCmd().
The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

11.124 CF_WriteQueueCmd Struct Reference

Write Queue command structure.

```
#include <default_cf_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CF_WriteQueue_Payload_t Payload`

11.124.1 Detailed Description

Write Queue command structure.

For command details see `CF_WRITE_QUEUE_CC`
Definition at line 225 of file default_cf_msgstruct.h.

11.124.2 Field Documentation

11.124.2.1 CommandHeader `CFE_MSG_CommandHeader_t CF_WriteQueueCmd::CommandHeader`
Command header.
Definition at line 227 of file default_cf_msgstruct.h.

11.124.2.2 Payload `CF_WriteQueue_Payload_t` `CF_WriteQueueCmd::Payload`

Definition at line 228 of file default_cf_msgstruct.h.

Referenced by `CF_WriteQueueCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

11.125 CFE_Config_ArrayValue Struct Reference

Wrapper type for array configuration.

```
#include <cfe_config_api_typedefs.h>
```

Data Fields

- `size_t NumElements`
- `const void * ElementPtr`

11.125.1 Detailed Description

Wrapper type for array configuration.

This is a pair containing a size and pointer that is get/set via a single config table entry

Definition at line 59 of file cfe_config_api_typedefs.h.

11.125.2 Field Documentation

11.125.2.1 ElementPtr `const void* CFE_Config_ArrayValue::ElementPtr`

Definition at line 62 of file cfe_config_api_typedefs.h.

11.125.2.2 NumElements `size_t CFE_Config_ArrayValue::NumElements`

Definition at line 61 of file cfe_config_api_typedefs.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h`

11.126 CFE_Config_IdNameEntry Struct Reference

```
#include <cfe_config_nametable.h>
```

Data Fields

- `const char * Name`

11.126.1 Detailed Description

Definition at line 33 of file cfe_config_nametable.h.

11.126.2 Field Documentation

11.126.2.1 Name `const char* CFE_Config_IdNameEntry::Name`

Definition at line 35 of file `cfe_config_nametable.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_nametable.h`

11.127 CFE_Config_ValueBuffer Union Reference

```
#include <cfe_config_table.h>
```

Data Fields

- `uint32 AsInteger`
- `const void * AsPointer`

11.127.1 Detailed Description

Definition at line 43 of file `cfe_config_table.h`.

11.127.2 Field Documentation**11.127.2.1 AsInteger** `uint32 CFE_Config_ValueBuffer::AsInteger`

Definition at line 45 of file `cfe_config_table.h`.

11.127.2.2 AsPointer `const void* CFE_Config_ValueBuffer::AsPointer`

Definition at line 46 of file `cfe_config_table.h`.

The documentation for this union was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_table.h`

11.128 CFE_Config_ValueEntry Struct Reference

```
#include <cfe_config_table.h>
```

Data Fields

- `CFE_ConfigType_t ActualType`
- `CFE_Config_ValueBuffer_t Datum`

11.128.1 Detailed Description

Definition at line 49 of file `cfe_config_table.h`.

11.128.2 Field Documentation**11.128.2.1 ActualType** `CFE_ConfigType_t CFE_Config_ValueEntry::ActualType`

Definition at line 51 of file `cfe_config_table.h`.

11.128.2.2 Datum `CFE_Config_ValueBuffer_t` `CFE_Config_ValueEntry::Datum`
Definition at line 52 of file `cfe_config_table.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_table.h`

11.129 CFE_ES_AppInfo Struct Reference

Application Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

Data Fields

- `CFE_ResourceId_t ResourceId`
Application or Library ID for this resource.
- `uint32 Type`
The type of App: CORE or EXTERNAL.
- `char Name [CFE_MISSION_MAX_API_LEN]`
The Registered Name of the Application.
- `char EntryPoint [CFE_MISSION_MAX_API_LEN]`
The Entry Point label for the Application.
- `char FileName [CFE_MISSION_MAX_PATH_LEN]`
The Filename of the file containing the Application.
- `CFE_ES_MemOffset_t StackSize`
The Stack Size of the Application.
- `uint32 AddressesAreValid`
Indicates that the Code, Data, and BSS addresses/sizes are valid.
- `CFE_ES_MemAddress_t CodeAddress`
The Address of the Application Code Segment.
- `CFE_ES_MemOffset_t CodeSize`
The Code Size of the Application.
- `CFE_ES_MemAddress_t DataAddress`
The Address of the Application Data Segment.
- `CFE_ES_MemOffset_t DataSize`
The Data Size of the Application.
- `CFE_ES_MemAddress_t BSSAddress`
The Address of the Application BSS Segment.
- `CFE_ES_MemOffset_t BSSSize`
The BSS Size of the Application.
- `CFE_ES_MemAddress_t StartAddress`
The Start Address of the Application.
- `CFE_ES_ExceptionAction_Enum_t ExceptionAction`
What should occur if Application has an exception (Restart Application OR Restart Processor)
- `CFE_ES_TaskPriority_Atom_t Priority`
The Priority of the Application.
- `CFE_ES_TaskId_t MainTaskId`
The Application's Main Task ID.
- `uint32 ExecutionCounter`
The Application's Main Task Execution Counter.

- char **MainTaskName** [CFE_MISSION_MAX_API_LEN]
The Application's Main Task ID.
- uint32 **NumOfChildTasks**
Number of Child tasks for an App.

11.129.1 Detailed Description

Application Information.

Structure that is used to provide information about an app. It is primarily used for the QueryOne and QueryAll Commands.

While this structure is primarily intended for Application info, it can also represent Library information where only a subset of the information applies.

Definition at line 441 of file default_cfe_es_extern_typedefs.h.

11.129.2 Field Documentation

11.129.2.1 AddressesAreValid `uint32 CFE_ES_AppInfo::AddressesAreValid`

Indicates that the Code, Data, and BSS addresses/sizes are valid.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_AddrsValid

Definition at line 457 of file default_cfe_es_extern_typedefs.h.

11.129.2.2 BSSAddress `CFE_ES_MemAddress_t CFE_ES_AppInfo::BSSAddress`

The Address of the Application BSS Segment.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BSSAddress

Definition at line 467 of file default_cfe_es_extern_typedefs.h.

11.129.2.3 BSSSize `CFE_ES_MemOffset_t CFE_ES_AppInfo::BSSSize`

The BSS Size of the Application.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BSSSize

Definition at line 469 of file default_cfe_es_extern_typedefs.h.

11.129.2.4 CodeAddress `CFE_ES_MemAddress_t CFE_ES_AppInfo::CodeAddress`

The Address of the Application Code Segment.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CodeAddress

Definition at line 459 of file default_cfe_es_extern_typedefs.h.

11.129.2.5 CodeSize `CFE_ES_MemOffset_t CFE_ES_AppInfo::CodeSize`

The Code Size of the Application.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CodeSize

Definition at line 461 of file default_cfe_es_extern_typedefs.h.

11.129.2.6 DataAddress `CFE_ES_MemAddress_t` `CFE_ES_AppInfo::DataAddress`
The Address of the Application Data Segment.

Telemetry Mnemonic(s) `$sc_$cpu_ES_DataAddress`

Definition at line 463 of file default_cfe_es_extern_typedefs.h.

11.129.2.7 DataSize `CFE_ES_MemOffset_t` `CFE_ES_AppInfo::DataSize`
The Data Size of the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_DataSize`

Definition at line 465 of file default_cfe_es_extern_typedefs.h.

11.129.2.8 EntryPoint `char` `CFE_ES_AppInfo::EntryPoint[CFE_MISSION_MAX_API_LEN]`
The Entry Point label for the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_AppEntryPt[OS_MAX_API_NAME]`

Definition at line 450 of file default_cfe_es_extern_typedefs.h.

11.129.2.9 ExceptionAction `CFE_ES_ExceptionAction_Enum_t` `CFE_ES_AppInfo::ExceptionAction`
What should occur if Application has an exception (Restart Application OR Restart Processor)

Telemetry Mnemonic(s) `$sc_$cpu_ES_ExceptnActn`

Definition at line 473 of file default_cfe_es_extern_typedefs.h.

11.129.2.10 ExecutionCounter `uint32` `CFE_ES_AppInfo::ExecutionCounter`
The Application's Main Task Execution Counter.

Telemetry Mnemonic(s) `$sc_$cpu_ES_ExecutionCtr`

Definition at line 480 of file default_cfe_es_extern_typedefs.h.

11.129.2.11 FileName `char` `CFE_ES_AppInfo::FileName[CFE_MISSION_MAX_PATH_LEN]`
The Filename of the file containing the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_AppFilename[OS_MAX_PATH_LEN]`

Definition at line 452 of file default_cfe_es_extern_typedefs.h.

11.129.2.12 MainTaskId `CFE_ES_TaskId_t` `CFE_ES_AppInfo::MainTaskId`
The Application's Main Task ID.

Telemetry Mnemonic(s) `$sc_$cpu_ES_MainTaskId`

Definition at line 478 of file default_cfe_es_extern_typedefs.h.

11.129.2.13 MainTaskName `char CFE_ES_AppInfo::MainTaskName [CFE_MISSION_MAX_API_LEN]`
The Application's Main Task ID.

Telemetry Mnemonic(s) `$sc_$cpu_ES_MainTaskName[OS_MAX_API_NAME]`

Definition at line 482 of file default_cfe_es_extern_typedefs.h.

11.129.2.14 Name `char CFE_ES_AppInfo::Name [CFE_MISSION_MAX_API_LEN]`
The Registered Name of the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_AppName[OS_MAX_API_NAME]`

Definition at line 448 of file default_cfe_es_extern_typedefs.h.

11.129.2.15 NumOfChildTasks `uint32 CFE_ES_AppInfo::NumOfChildTasks`
Number of Child tasks for an App.

Telemetry Mnemonic(s) `$sc_$cpu_ES_ChildTasks`

Definition at line 484 of file default_cfe_es_extern_typedefs.h.

11.129.2.16 Priority `CFE_ES_TaskPriority_Atom_t CFE_ES_AppInfo::Priority`
The Priority of the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_Priority`

Definition at line 476 of file default_cfe_es_extern_typedefs.h.

11.129.2.17 ResourceId `CFE_ResourceId_t CFE_ES_AppInfo::ResourceId`
Application or Library ID for this resource.

Telemetry Mnemonic(s) `$sc_$cpu_ES_AppID`

Definition at line 443 of file default_cfe_es_extern_typedefs.h.

11.129.2.18 StackSize `CFE_ES_MemOffset_t CFE_ES_AppInfo::StackSize`
The Stack Size of the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_StackSize`

Definition at line 455 of file default_cfe_es_extern_typedefs.h.

11.129.2.19 StartAddress `CFE_ES_MemAddress_t CFE_ES_AppInfo::StartAddress`
The Start Address of the Application.

Telemetry Mnemonic(s) `$sc_$cpu_ES_StartAddr`

Definition at line 471 of file default_cfe_es_extern_typedefs.h.

11.129.2.20 Type `uint32 CFE_ES_AppInfo::Type`

The type of App: CORE or EXTERNAL.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_AppType

Definition at line 445 of file default_cfe_es_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_extern_typedefs.h

11.130 CFE_ES_AppNameCmd_Payload Struct Reference

Generic application name command payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- char `Application [CFE_MISSION_MAX_API_LEN]`

ASCII text string containing Application or Library Name.

11.130.1 Detailed Description

Generic application name command payload.

For command details, see [CFE_ES_STOP_APP_CC](#), [CFE_ES_RESTART_APP_CC](#), [CFE_ES_QUERY_ONE_CC](#)
Definition at line 104 of file default_cfe_es_msgdefs.h.

11.130.2 Field Documentation

11.130.2.1 Application `char CFE_ES_AppNameCmd_Payload::Application [CFE_MISSION_MAX_API_LEN]`

ASCII text string containing Application or Library Name.

Definition at line 106 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.131 CFE_ES_AppReloadCmd_Payload Struct Reference

Reload Application Command Payload.

```
#include <default_cfe_es_msgdefs.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.

11.131.1 Detailed Description

Reload Application Command Payload.

For command details, see [CFE_ES_RELOAD_APP_CC](#)

Definition at line 115 of file default_cfe_es_msgdefs.h.

11.131.2 Field Documentation

11.131.2.1 AppFileName `char CFE_ES_AppReloadCmd_Payload::AppFileName[CFE_MISSION_MAX_PATH_LEN]`

Full path and filename of Application's executable image.

Definition at line 118 of file `default_cfe_es_msgdefs.h`.

11.131.2.2 Application `char CFE_ES_AppReloadCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]`

ASCII text string containing Application Name.

Definition at line 117 of file `default_cfe_es_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgdefs.h`

11.132 CFE_ES_BlockStats Struct Reference

Block statistics.

```
#include <default_cfe_es_extern_typedefs.h>
```

Data Fields

- `CFE_ES_MemOffset_t 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.

11.132.1 Detailed Description

Block statistics.

Sub-Structure that is used to provide information about a specific block size/bucket within a memory pool.

Definition at line 538 of file `default_cfe_es_extern_typedefs.h`.

11.132.2 Field Documentation

11.132.2.1 BlockSize `CFE_ES_MemOffset_t CFE_ES_BlockStats::BlockSize`

Number of bytes in each of these blocks.

Definition at line 540 of file `default_cfe_es_extern_typedefs.h`.

11.132.2.2 NumCreated `uint32 CFE_ES_BlockStats::NumCreated`

Number of Memory Blocks of this size created.

Definition at line 541 of file `default_cfe_es_extern_typedefs.h`.

11.132.2.3 NumFree `uint32` `CFE_ES_BlockStats::NumFree`

Number of Memory Blocks of this size that are free.

Definition at line 542 of file `default_cfe_es_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_extern_typedefs.h`

11.133 CFE_ES_CDSRegDumpRec Struct Reference

CDS Register Dump Record.

```
#include <default_cfe_es_extern_typedefs.h>
```

Data Fields

- **`CFE_ES_CDSHandle_t Handle`**
Handle of CDS.
- **`CFE_ES_MemOffset_t Size`**
Size, in bytes, of the CDS memory block.
- `bool Table`
Flag that indicates whether CDS contains a Critical Table.
- `char Name [CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`
Processor Unique Name of CDS.
- `uint8 ByteAlignSpare [3]`
Spare bytes to ensure structure size is multiple of 4 bytes.

11.133.1 Detailed Description

CDS Register Dump Record.

Structure that is used to provide information about a critical data store. It is primarily used for the Dump CDS registry (`CFE_ES_DUMP_CDS_REGISTRY_CC`) command.

Note

There is not currently a telemetry message directly containing this data structure, but it does define the format of the data file generated by the Dump CDS registry command. Therefore it should be considered part of the overall telemetry interface.

Definition at line 523 of file `default_cfe_es_extern_typedefs.h`.

11.133.2 Field Documentation

11.133.2.1 **ByteAlignSpare** `uint8` `CFE_ES_CDSRegDumpRec::ByteAlignSpare[3]`

Spare bytes to ensure structure size is multiple of 4 bytes.

Definition at line 529 of file `default_cfe_es_extern_typedefs.h`.

11.133.2.2 **Handle** `CFE_ES_CDSHandle_t` `CFE_ES_CDSRegDumpRec::Handle`

Handle of CDS.

Definition at line 525 of file `default_cfe_es_extern_typedefs.h`.

11.133.2.3 Name `char CFE_ES_CDSRegDumpRec::Name[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`
Processor Unique Name of CDS.
Definition at line 528 of file default_cfe_es_extern_typedefs.h.

11.133.2.4 Size `CFE_ES_MemOffset_t CFE_ES_CDSRegDumpRec::Size`
Size, in bytes, of the CDS memory block.
Definition at line 526 of file default_cfe_es_extern_typedefs.h.

11.133.2.5 Table `bool CFE_ES_CDSRegDumpRec::Table`
Flag that indicates whether CDS contains a Critical Table.
Definition at line 527 of file default_cfe_es_extern_typedefs.h.
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_extern_typedefs.h

11.134 CFE_ES_ClearERLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.134.1 Detailed Description

Definition at line 68 of file default_cfe_es_msgstruct.h.

11.134.2 Field Documentation

11.134.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_ClearERLogCmd::CommandHeader`
Command header.
Definition at line 70 of file default_cfe_es_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.135 CFE_ES_ClearSysLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.135.1 Detailed Description

Definition at line 63 of file default_cfe_es_msgstruct.h.

11.135.2 Field Documentation

11.135.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_ClearSysLogCmd::CommandHeader
Command header.

Definition at line 65 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.136 CFE_ES_DeleteCDSCmd Struct Reference

Delete Critical Data Store Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_DeleteCDSCmd_Payload_t](#) Payload
Command payload.

11.136.1 Detailed Description

Delete Critical Data Store Command.

Definition at line 192 of file default_cfe_es_msgstruct.h.

11.136.2 Field Documentation

11.136.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_DeleteCDSCmd::CommandHeader
Command header.

Definition at line 194 of file default_cfe_es_msgstruct.h.

11.136.2.2 Payload [CFE_ES_DeleteCDSCmd_Payload_t](#) CFE_ES_DeleteCDSCmd::Payload
Command payload.

Definition at line 195 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.137 CFE_ES_DeleteCDSCmd_Payload Struct Reference

Delete Critical Data Store Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- char [CdsName](#) [[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN](#)]
ASCII text string containing name of CDS to delete.

11.137.1 Detailed Description

Delete Critical Data Store Command Payload.
For command details, see [CFE_ES_DELETE_CDS_CC](#)
Definition at line 140 of file default_cfe_es_msgdefs.h.

11.137.2 Field Documentation

11.137.2.1 CdsName `char CFE_ES_DeleteCDSCmd_Payload::CdsName[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`
ASCII text string containing name of CDS to delete.

Definition at line 143 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgdefs.h](#)

11.138 CFE_ES_DumpCDSRegistryCmd Struct Reference

Dump CDS Registry Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_ES_DumpCDSRegistryCmd_Payload_t Payload](#)
Command payload.

11.138.1 Detailed Description

Dump CDS Registry Command.

Definition at line 246 of file default_cfe_es_msgstruct.h.

11.138.2 Field Documentation

11.138.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) `CFE_ES_DumpCDSRegistryCmd::CommandHeader`
Command header.

Definition at line 248 of file default_cfe_es_msgstruct.h.

11.138.2.2 Payload [CFE_ES_DumpCDSRegistryCmd_Payload_t](#) `CFE_ES_DumpCDSRegistryCmd::Payload`
Command payload.

Definition at line 249 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.139 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference

Dump CDS Registry Command Payload.

```
#include <default_cfe_es_msgdefs.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.

11.139.1 Detailed Description

Dump CDS Registry Command Payload.

For command details, see [CFE_ES_DUMP_CDS_REGISTRY_CC](#)

Definition at line 225 of file default_cfe_es_msgdefs.h.

11.139.2 Field Documentation

11.139.2.1 DumpFilename `char CFE_ES_DumpCDSRegistryCmd_Payload::DumpFilename[CFE_MISSION_MAX_PATH_LEN]`

ASCII text string of full path and filename of file CDS Registry is to be written.

Definition at line 227 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.140 CFE_ES_FileNameCmd Struct Reference

Generic file name command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_ES_FileNameCmd_Payload_t Payload`
Command payload.

11.140.1 Detailed Description

Generic file name command.

Definition at line 96 of file default_cfe_es_msgstruct.h.

11.140.2 Field Documentation

11.140.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_FileNameCmd::CommandHeader`

Command header.

Definition at line 98 of file default_cfe_es_msgstruct.h.

11.140.2.2 Payload `CFE_ES_FileNameCmd_Payload_t CFE_ES_FileNameCmd::Payload`

Command payload.

Definition at line 99 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.141 CFE_ES_FileNameCmd_Payload Struct Reference

Generic file name command payload.

```
#include <default_cfe_es_msgdefs.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.

11.141.1 Detailed Description

Generic file name command payload.

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_SYS_LOG_CC](#), and [CFE_ES_WRITE_ER_LOG_CC](#).
Definition at line 58 of file default_cfe_es_msgdefs.h.

11.141.2 Field Documentation

11.141.2.1 FileName char CFE_ES_FileNameCmd_Payload::FileName[[CFE_MISSION_MAX_PATH_LEN](#)]

ASCII text string containing full path and filename of file in which Application data is to be dumped.

Definition at line 60 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/[default_cfe_es_msgdefs.h](#)

11.142 CFE_ES_HousekeepingTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_ES_HousekeepingTlm_Payload_t](#) Payload
Telemetry payload.

11.142.1 Detailed Description

Name Executive Services Housekeeping Packet

Definition at line 279 of file default_cfe_es_msgstruct.h.

11.142.2 Field Documentation

11.142.2.1 Payload [CFE_ES_HousekeepingTlm_Payload_t](#) CFE_ES_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 282 of file default_cfe_es_msgstruct.h.

11.142.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_ES_HousekeepingTlm::TelemetryHeader`
Telemetry header.

Definition at line 281 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

11.143 CFE_ES_HousekeepingTlm_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.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.
- `uint8 PSPMajorVersion`
Platform Support Package Major Version Number.
- `uint8 PSPMinorVersion`
Platform Support Package Minor Version Number.
- `uint8 PSPRevision`
Platform Support Package Revision Number.
- `uint8 PSPMissionRevision`
Platform Support Package MissionRevision Number.
- `CFE_ES_MemOffset_t SysLogBytesUsed`
Total number of bytes used in system log.
- `CFE_ES_MemOffset_t 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 Performance 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.
- **CFE_ES_MemOffset_t HeapBytesFree**
Number of free bytes remaining in the OS heap.
- **CFE_ES_MemOffset_t HeapBlocksFree**
Number of free blocks remaining in the OS heap.
- **CFE_ES_MemOffset_t HeapMaxBlockSize**
Number of bytes in the largest free block.

11.143.1 Detailed Description

Name Executive Services Housekeeping Packet

Definition at line 263 of file default_cfe_es_msgdefs.h.

11.143.2 Field Documentation

11.143.2.1 BootSource `uint32` `CFE_ES_HousekeepingTlm_Payload::BootSource`
Boot source (as provided from BSP)

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BootSource

Definition at line 329 of file default_cfe_es_msgdefs.h.

11.143.2.2 CFECOREChecksum `uint16` `CFE_ES_HousekeepingTlm_Payload::CFECOREChecksum`
Checksum of cFE Core Code.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CKSUM

Definition at line 270 of file default_cfe_es_msgdefs.h.

11.143.2.3 CFEMajorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::CFEMajorVersion`
Major Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMAJORVER

Definition at line 272 of file default_cfe_es_msgdefs.h.

11.143.2.4 CFEMinorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::CFEMinorVersion`
Minor Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMINORVER

Definition at line 274 of file default_cfe_es_msgdefs.h.

11.143.2.5 CFEMissionRevision `uint8` `CFE_ES_HousekeepingTlm_Payload::CFEMissionRevision`
Mission Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMISSIONREV

Definition at line 278 of file default_cfe_es_msgdefs.h.

11.143.2.6 CFERevision `uint8` `CFE_ES_HousekeepingTlm_Payload::CFERevision`
Sub-Minor Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEREVISION

Definition at line 276 of file default_cfe_es_msgdefs.h.

11.143.2.7 CommandCounter `uint8 CFE_ES_HousekeepingTlm_Payload::CommandCounter`
The ES Application Command Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CMDPC

Definition at line 265 of file default_cfe_es_msgdefs.h.

11.143.2.8 CommandErrorCounter `uint8 CFE_ES_HousekeepingTlm_Payload::CommandErrorCounter`
The ES Application Command Error Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CMDEC

Definition at line 267 of file default_cfe_es_msgdefs.h.

11.143.2.9 ERLogEntries `uint32 CFE_ES_HousekeepingTlm_Payload::ERLogEntries`
Number of entries made in the ER Log since the power on.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ERLOGENTRIES

Definition at line 309 of file default_cfe_es_msgdefs.h.

11.143.2.10 ERLogIndex `uint32 CFE_ES_HousekeepingTlm_Payload::ERLogIndex`
Current index of the ER Log (wraps around)

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ERLOGINDEX

Definition at line 307 of file default_cfe_es_msgdefs.h.

11.143.2.11 HeapBlocksFree `CFE_ES_MemOffset_t CFE_ES_HousekeepingTlm_Payload::HeapBlocksFree`
Number of free blocks remaining in the OS heap.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_HeapBlocksFree

Definition at line 354 of file default_cfe_es_msgdefs.h.

11.143.2.12 HeapBytesFree `CFE_ES_MemOffset_t CFE_ES_HousekeepingTlm_Payload::HeapBytesFree`
Number of free bytes remaining in the OS heap.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_HeapBytesFree

Definition at line 352 of file default_cfe_es_msgdefs.h.

11.143.2.13 HeapMaxBlockSize `CFE_ES_MemOffset_t CFE_ES_HousekeepingTlm_Payload::HeapMaxBlockSize`
Number of bytes in the largest free block.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_HeapMaxBlkSize

Definition at line 356 of file default_cfe_es_msgdefs.h.

11.143.2.14 MaxProcessorResets `uint32` `CFE_ES_HousekeepingTlm_Payload::MaxProcessorResets`
Max processor resets before a power on is done.

Telemetry Mnemonic(s) `$sc_$cpu_ES_MaxProcResets`

Definition at line 327 of file `default_cfe_es_msgdefs.h`.

11.143.2.15 OSALMajorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMajorVersion`
OS Abstraction Layer Major Version Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_OSMajorVer`

Definition at line 280 of file `default_cfe_es_msgdefs.h`.

11.143.2.16 OSALMinorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMinorVersion`
OS Abstraction Layer Minor Version Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_Osminorver`

Definition at line 282 of file `default_cfe_es_msgdefs.h`.

11.143.2.17 OSALMissionRevision `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMissionRevision`
OS Abstraction Layer MissionRevision Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_Osmissionrev`

Definition at line 286 of file `default_cfe_es_msgdefs.h`.

11.143.2.18 OSALRevision `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALRevision`
OS Abstraction Layer Revision Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_Osrevision`

Definition at line 284 of file `default_cfe_es_msgdefs.h`.

11.143.2.19 PerfDataCount `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataCount`
Number of Entries Put Into the Performance Analyzer Log.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PerfDataCnt`

Definition at line 347 of file `default_cfe_es_msgdefs.h`.

11.143.2.20 PerfDataEnd `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataEnd`
Identifies Last Stored Entry in Performance Analyzer Log.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PerfDataEnd`

Definition at line 345 of file `default_cfe_es_msgdefs.h`.

11.143.2.21 PerfDataStart `uint32 CFE_ES_HousekeepingTlm_Payload::PerfDataStart`
Identifies First Stored Entry in Performance Analyzer Log.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfDataStart

Definition at line 343 of file default_cfe_es_msgdefs.h.

11.143.2.22 PerfDataToWrite `uint32 CFE_ES_HousekeepingTlm_Payload::PerfDataToWrite`
Number of Performance Analyzer Log Entries Left to be Written to Log Dump File.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfData2Write

Definition at line 350 of file default_cfe_es_msgdefs.h.

11.143.2.23 PerfFilterMask `uint32 CFE_ES_HousekeepingTlm_Payload::PerfFilterMask[CFE_MISSION_ES_PERF_MAX_IDS/32]`
Current Setting of Performance Analyzer Filter Masks.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfFltrMask[MaskCnt]

Definition at line 338 of file default_cfe_es_msgdefs.h.

11.143.2.24 PerfMode `uint32 CFE_ES_HousekeepingTlm_Payload::PerfMode`
Current mode of Performance Analyzer.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfMode

Definition at line 334 of file default_cfe_es_msgdefs.h.

11.143.2.25 PerfState `uint32 CFE_ES_HousekeepingTlm_Payload::PerfState`
Current state of Performance Analyzer.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfState

Definition at line 332 of file default_cfe_es_msgdefs.h.

11.143.2.26 PerfTriggerCount `uint32 CFE_ES_HousekeepingTlm_Payload::PerfTriggerCount`
Number of Times Performance Analyzer has Triggered.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfTrigCnt

Definition at line 336 of file default_cfe_es_msgdefs.h.

11.143.2.27 PerfTriggerMask `uint32 CFE_ES_HousekeepingTlm_Payload::PerfTriggerMask[CFE_MISSION_ES_PERF_MAX_IDS/32]`
Current Setting of Performance Analyzer Trigger Masks.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfTrigMask[MaskCnt]

Definition at line 341 of file default_cfe_es_msgdefs.h.

11.143.2.28 ProcessorResets `uint32` `CFE_ES_HousekeepingTlm_Payload::ProcessorResets`
Number of processor resets since last power on.

Telemetry Mnemonic(s) `$sc_$cpu_ES_ProcResetCnt`

Definition at line 325 of file `default_cfe_es_msgdefs.h`.

11.143.2.29 PSPMajorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion`
Platform Support Package Major Version Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PSPMAJORVER`

Definition at line 289 of file `default_cfe_es_msgdefs.h`.

11.143.2.30 PSPMinorVersion `uint8` `CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion`
Platform Support Package Minor Version Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PSPMINORVER`

Definition at line 291 of file `default_cfe_es_msgdefs.h`.

11.143.2.31 PSPMissionRevision `uint8` `CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision`
Platform Support Package MissionRevision Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PSPMISSIONREV`

Definition at line 295 of file `default_cfe_es_msgdefs.h`.

11.143.2.32 PSPRevision `uint8` `CFE_ES_HousekeepingTlm_Payload::PSPRevision`
Platform Support Package Revision Number.

Telemetry Mnemonic(s) `$sc_$cpu_ES_PSPREVISION`

Definition at line 293 of file `default_cfe_es_msgdefs.h`.

11.143.2.33 RegisteredCoreApps `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps`
Number of Applications registered with ES.

Telemetry Mnemonic(s) `$sc_$cpu_ES_RegCoreApps`

Definition at line 312 of file `default_cfe_es_msgdefs.h`.

11.143.2.34 RegisteredExternalApps `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps`
Number of Applications registered with ES.

Telemetry Mnemonic(s) `$sc_$cpu_ES_RegExtApps`

Definition at line 314 of file `default_cfe_es_msgdefs.h`.

11.143.2.35 RegisteredLibs `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredLibs`
Number of Libraries registered with ES.

Telemetry Mnemonic(s) `$sc_$cpu_ES_RegLibs`

Definition at line 318 of file `default_cfe_es_msgdefs.h`.

11.143.2.36 RegisteredTasks `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredTasks`
Number of Tasks (main AND child tasks) registered with ES.

Telemetry Mnemonic(s) `$sc_$cpu_ES_RegTasks`

Definition at line 316 of file `default_cfe_es_msgdefs.h`.

11.143.2.37 ResetSubtype `uint32` `CFE_ES_HousekeepingTlm_Payload::ResetSubtype`
Reset Sub Type.

Telemetry Mnemonic(s) `$sc_$cpu_ES_ResetSubtype`

Definition at line 323 of file `default_cfe_es_msgdefs.h`.

11.143.2.38 ResetType `uint32` `CFE_ES_HousekeepingTlm_Payload::ResetType`
Reset type (PROCESSOR or POWERON)

Telemetry Mnemonic(s) `$sc_$cpu_ES_ResetType`

Definition at line 321 of file `default_cfe_es_msgdefs.h`.

11.143.2.39 SysLogBytesUsed `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed`
Total number of bytes used in system log.

Telemetry Mnemonic(s) `$sc_$cpu_ES_SYSLOGBYTEUSED`

Definition at line 298 of file `default_cfe_es_msgdefs.h`.

11.143.2.40 SysLogEntries `uint32` `CFE_ES_HousekeepingTlm_Payload::SysLogEntries`
Number of entries in the system log.

Telemetry Mnemonic(s) `$sc_$cpu_ES_SYSLOGENTRIES`

Definition at line 302 of file `default_cfe_es_msgdefs.h`.

11.143.2.41 SysLogMode `uint32` `CFE_ES_HousekeepingTlm_Payload::SysLogMode`
Write/Overwrite Mode.

Telemetry Mnemonic(s) `$sc_$cpu_ES_SYSLOGMODE`

Definition at line 304 of file `default_cfe_es_msgdefs.h`.

11.143.2.42 SysLogSize [CFE_ES_MemOffset_t](#) CFE_ES_HousekeepingTlm_Payload::SysLogSize
Total size of the system log.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_SYSLOGSIZE

Definition at line 300 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.144 CFE_ES_MemPoolStats Struct Reference

Memory Pool Statistics.

```
#include <default_cfe_es_extern_typedefs.h>
```

Data Fields

- [CFE_ES_MemOffset_t](#) 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.
- [CFE_ES_MemOffset_t](#) NumFreeBytes
Number of bytes never allocated to a block.
- [CFE_ES_BlockStats_t](#) BlockStats [[CFE_MISSION_ES_POOL_MAX_BUCKETS](#)]
Contains stats on each block size.

11.144.1 Detailed Description

Memory Pool Statistics.

Structure that is used to provide information about a memory pool. Used by the Memory Pool Stats telemetry message.

See also

[CFE_ES_SEND_MEM_POOL_STATS_CC](#)

Definition at line 553 of file default_cfe_es_extern_typedefs.h.

11.144.2 Field Documentation

11.144.2.1 BlockStats [CFE_ES_BlockStats_t](#) CFE_ES_MemPoolStats::BlockStats [[CFE_MISSION_ES_POOL_MAX_BUCKETS](#)]
Contains stats on each block size.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BlkStats[BLK_SIZES]

Definition at line 563 of file default_cfe_es_extern_typedefs.h.

11.144.2.2 CheckErrCtr [uint32](#) CFE_ES_MemPoolStats::CheckErrCtr
Number of errors detected when freeing a memory block.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BlkErrCTR

Definition at line 559 of file default_cfe_es_extern_typedefs.h.

11.144.2.3 NumBlocksRequested `uint32 CFE_ES_MemPoolStats::NumBlocksRequested`
Number of times a memory block has been allocated.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BlkREQ

Definition at line 557 of file default_cfe_es_extern_typedefs.h.

11.144.2.4 NumFreeBytes `CFE_ES_MemOffset_t CFE_ES_MemPoolStats::NumFreeBytes`
Number of bytes never allocated to a block.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_FreeBytes

Definition at line 561 of file default_cfe_es_extern_typedefs.h.

11.144.2.5 PoolSize `CFE_ES_MemOffset_t CFE_ES_MemPoolStats::PoolSize`
Size of Memory Pool (in bytes)

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PoolSize

Definition at line 555 of file default_cfe_es_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_extern_typedefs.h

11.145 CFE_ES_MemStatsTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_ES_PoolStatsTlm_Payload_t Payload`
Telemetry payload.

11.145.1 Detailed Description

Name Memory Pool Statistics Packet

Definition at line 270 of file default_cfe_es_msgstruct.h.

11.145.2 Field Documentation

11.145.2.1 Payload `CFE_ES_PoolStatsTlm_Payload_t CFE_ES_MemStatsTlm::Payload`
Telemetry payload.

Definition at line 273 of file default_cfe_es_msgstruct.h.

11.145.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t CFE_ES_MemStatsTlm::TelemetryHeader`
Telemetry header.

Definition at line 272 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.146 CFE_ES_NoopCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

11.146.1 Detailed Description

Definition at line 53 of file default_cfe_es_msgstruct.h.

11.146.2 Field Documentation

11.146.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_NoopCmd::CommandHeader

Command header.

Definition at line 57 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.147 CFE_ES_OneAppTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader

Telemetry header.

- [CFE_ES_OneAppTlm_Payload_t](#) Payload

Telemetry payload.

11.147.1 Detailed Description

Name Single Application Information Packet

Definition at line 261 of file default_cfe_es_msgstruct.h.

11.147.2 Field Documentation

11.147.2.1 Payload [CFE_ES_OneAppTlm_Payload_t](#) CFE_ES_OneAppTlm::Payload

Telemetry payload.

Definition at line 264 of file default_cfe_es_msgstruct.h.

11.147.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CFE_ES_OneAppTlm::TelemetryHeader

Telemetry header.

Definition at line 263 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.148 CFE_ES_OneAppTlm_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [CFE_ES_AppInfo_t AppInfo](#)

For more information, see [CFE_ES_AppInfo_t](#).

11.148.1 Detailed Description

Name Single Application Information Packet

Definition at line 243 of file default_cfe_es_msgdefs.h.

11.148.2 Field Documentation

11.148.2.1 AppInfo [CFE_ES_AppInfo_t](#) CFE_ES_OneAppTlm_Payload::AppInfo

For more information, see [CFE_ES_AppInfo_t](#).

Definition at line 245 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.149 CFE_ES_OverWriteSysLogCmd Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_ES_OverWriteSysLogCmd_Payload_t Payload](#)
Command payload.

11.149.1 Detailed Description

Overwrite/Discard System Log Configuration Command Payload.

Definition at line 133 of file default_cfe_es_msgstruct.h.

11.149.2 Field Documentation

11.149.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_OverWriteSysLogCmd::CommandHeader

Command header.

Definition at line 135 of file default_cfe_es_msgstruct.h.

11.149.2.2 Payload `CFE_ES_OverWriteSysLogCmd_Payload_t` `CFE_ES_OverWriteSysLogCmd::Payload`
Command payload.

Definition at line 136 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.150 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- `uint32 Mode`

`CFE_ES_LogMode_DISCARD`=Throw away most recent messages, `CFE_ES_LogMode_OVERWRITE`=Overwrite oldest with most recent

11.150.1 Detailed Description

Overwrite/Discard System Log Configuration Command Payload.

For command details, see [CFE_ES_OVER_WRITE_SYS_LOG_CC](#)

Definition at line 70 of file default_cfe_es_msgdefs.h.

11.150.2 Field Documentation

11.150.2.1 Mode `uint32 CFE_ES_OverWriteSysLogCmd_Payload::Mode`

`CFE_ES_LogMode_DISCARD`=Throw away most recent messages, `CFE_ES_LogMode_OVERWRITE`=Overwrite oldest with most recent

Definition at line 72 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.151 CFE_ES_PoolAlign Union Reference

Pool Alignment.

```
#include <cfe_es_api_typedefs.h>
```

Data Fields

- `void * Ptr`

Aligned pointer.

- `long long int LongInt`

Aligned Long Integer.

- `long double LongDouble`

Aligned Long Double.

11.151.1 Detailed Description

Pool Alignment.

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 145 of file cfe_es_api_typedefs.h.

11.151.2 Field Documentation

11.151.2.1 LongDouble long double CFE_ES_PoolAlign::LongDouble

Aligned Long Double.

Definition at line 150 of file cfe_es_api_typedefs.h.

11.151.2.2 LongInt long long int CFE_ES_PoolAlign::LongInt

Aligned Long Integer.

Definition at line 149 of file cfe_es_api_typedefs.h.

11.151.2.3 Ptr void* CFE_ES_PoolAlign::Ptr

Aligned pointer.

Definition at line 147 of file cfe_es_api_typedefs.h.

The documentation for this union was generated from the following file:

- cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h

11.152 CFE_ES_PoolStatsTlm_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.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](#).

11.152.1 Detailed Description

Name Memory Pool Statistics Packet

Definition at line 251 of file default_cfe_es_msgdefs.h.

11.152.2 Field Documentation

11.152.2.1 PoolHandle [CFE_ES_MemHandle_t](#) CFE_ES_PoolStatsTlm_Payload::PoolHandle

Handle of memory pool whose stats are being telemetered.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PoolHandle

Definition at line 253 of file default_cfe_es_msgdefs.h.

11.152.2.2 PoolStats [CFE_ES_MemPoolStats_t](#) CFE_ES_PoolStatsTlm_Payload::PoolStats

For more info, see [CFE_ES_MemPoolStats_t](#).

Definition at line 255 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.153 CFE_ES_QueryAllCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_FileNameCmd_Payload_t](#) Payload
Command payload.

11.153.1 Detailed Description

Definition at line 106 of file default_cfe_es_msgstruct.h.

11.153.2 Field Documentation**11.153.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_ES_QueryAllCmd::CommandHeader

Command header.

Definition at line 108 of file default_cfe_es_msgstruct.h.

11.153.2.2 Payload [CFE_ES_FileNameCmd_Payload_t](#) CFE_ES_QueryAllCmd::Payload

Command payload.

Definition at line 109 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.154 CFE_ES_QueryAllTasksCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_FileNameCmd_Payload_t](#) Payload
Command payload.

11.154.1 Detailed Description

Definition at line 112 of file default_cfe_es_msgstruct.h.

11.154.2 Field Documentation

11.154.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_ES_QueryAllTasksCmd::CommandHeader`
Command header.

Definition at line 114 of file `default_cfe_es_msgstruct.h`.

11.154.2.2 Payload `CFE_ES_FileNameCmd_Payload_t` `CFE_ES_QueryAllTasksCmd::Payload`
Command payload.

Definition at line 115 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.155 CFE_ES_QueryOneCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** `CommandHeader`
Command header.
- **CFE_ES_AppNameCmd_Payload_t** `Payload`
Command payload.

11.155.1 Detailed Description

Definition at line 165 of file `default_cfe_es_msgstruct.h`.

11.155.2 Field Documentation

11.155.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_ES_QueryOneCmd::CommandHeader`
Command header.

Definition at line 167 of file `default_cfe_es_msgstruct.h`.

11.155.2.2 Payload `CFE_ES_AppNameCmd_Payload_t` `CFE_ES_QueryOneCmd::Payload`
Command payload.

Definition at line 168 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.156 CFE_ES_ReloadAppCmd Struct Reference

Reload Application Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_AppReloadCmd_Payload_t](#) Payload
Command payload.

11.156.1 Detailed Description

Reload Application Command.

Definition at line 174 of file default_cfe_es_msgstruct.h.

11.156.2 Field Documentation**11.156.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_ES_ReloadAppCmd::CommandHeader
Command header.

Definition at line 176 of file default_cfe_es_msgstruct.h.

11.156.2.2 Payload [CFE_ES_AppReloadCmd_Payload_t](#) CFE_ES_ReloadAppCmd::Payload
Command payload.

Definition at line 177 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.157 CFE_ES_ResetCountersCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.157.1 Detailed Description

Definition at line 58 of file default_cfe_es_msgstruct.h.

11.157.2 Field Documentation**11.157.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_ES_ResetCountersCmd::CommandHeader
Command header.

Definition at line 60 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.158 CFE_ES_ResetPRCountCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.158.1 Detailed Description

Definition at line 73 of file default_cfe_es_msgstruct.h.

11.158.2 Field Documentation**11.158.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_ES_ResetPRCountCmd::CommandHeader
Command header.

Definition at line 75 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/[default_cfe_es_msgstruct.h](#)

11.159 CFE_ES_RestartAppCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_AppNameCmd_Payload_t](#) Payload
Command payload.

11.159.1 Detailed Description

Definition at line 159 of file default_cfe_es_msgstruct.h.

11.159.2 Field Documentation**11.159.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_ES_RestartAppCmd::CommandHeader
Command header.

Definition at line 161 of file default_cfe_es_msgstruct.h.

11.159.2.2 Payload [CFE_ES_AppNameCmd_Payload_t](#) CFE_ES_RestartAppCmd::Payload
Command payload.

Definition at line 162 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/[default_cfe_es_msgstruct.h](#)

11.160 CFE_ES_RestartCmd Struct Reference

Restart cFE Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_RestartCmd_Payload_t](#) Payload
Command payload.

11.160.1 Detailed Description

Restart cFE Command.

Definition at line 86 of file default_cfe_es_msgstruct.h.

11.160.2 Field Documentation**11.160.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_RestartCmd::CommandHeader**
Command header.

Definition at line 88 of file default_cfe_es_msgstruct.h.

11.160.2.2 Payload [CFE_ES_RestartCmd_Payload_t](#) CFE_ES_RestartCmd::Payload
Command payload.

Definition at line 89 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.161 CFE_ES_RestartCmd_Payload Struct Reference

Restart cFE Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [uint16 RestartType](#)
CFE_PSP_RST_TYPE_PROCESSOR=Processor Reset or CFE_PSP_RST_TYPE_POWERON=Power-On Reset

11.161.1 Detailed Description

Restart cFE Command Payload.

For command details, see [CFE_ES_RESTART_CC](#)

Definition at line 44 of file default_cfe_es_msgdefs.h.

11.161.2 Field Documentation**11.161.2.1 RestartType [uint16](#) CFE_ES_RestartCmd_Payload::RestartType**
CFE_PSP_RST_TYPE_PROCESSOR=Processor Reset or CFE_PSP_RST_TYPE_POWERON=Power-On Reset

Definition at line 46 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgdefs.h](#)

11.162 CFE_ES_SendHkCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

11.162.1 Detailed Description

Definition at line 78 of file default_cfe_es_msgstruct.h.

11.162.2 Field Documentation

11.162.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_SendHkCmd::CommandHeader

Command header.

Definition at line 80 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/[default_cfe_es_msgstruct.h](#)

11.163 CFE_ES_SendMemPoolStatsCmd Struct Reference

Send Memory Pool Statistics Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

- [CFE_ES_SendMemPoolStatsCmd_Payload_t](#) Payload

Command payload.

11.163.1 Detailed Description

Send Memory Pool Statistics Command.

Definition at line 237 of file default_cfe_es_msgstruct.h.

11.163.2 Field Documentation

11.163.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_SendMemPoolStatsCmd::CommandHeader

Command header.

Definition at line 239 of file default_cfe_es_msgstruct.h.

11.163.2.2 Payload [CFE_ES_SendMemPoolStatsCmd_Payload_t](#) CFE_ES_SendMemPoolStatsCmd::Payload

Command payload.

Definition at line 240 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/[default_cfe_es_msgstruct.h](#)

11.164 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference

Send Memory Pool Statistics Command Payload.

```
#include <default_cfe_es_msgdefs.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.

11.164.1 Detailed Description

Send Memory Pool Statistics Command Payload.

For command details, see [CFE_ES_SEND_MEM_POOL_STATS_CC](#)

Definition at line 213 of file default_cfe_es_msgdefs.h.

11.164.2 Field Documentation

11.164.2.1 Application `char CFE_ES_SendMemPoolStatsCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]`

- RESERVED - should be all zeroes

Definition at line 215 of file default_cfe_es_msgdefs.h.

11.164.2.2 PoolHandle `CFE_ES_MemHandle_t CFE_ES_SendMemPoolStatsCmd_Payload::PoolHandle`

Handle of Pool whose statistics are to be telemetered.

Definition at line 216 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.165 CFE_ES_SetMaxPRCountCmd Struct Reference

Set Maximum Processor Reset Count Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_ES_SetMaxPRCountCmd_Payload_t Payload](#)
Command payload.

11.165.1 Detailed Description

Set Maximum Processor Reset Count Command.

Definition at line 183 of file default_cfe_es_msgstruct.h.

11.165.2 Field Documentation

11.165.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_SetMaxPRCountCmd::CommandHeader
Command header.

Definition at line 185 of file default_cfe_es_msgstruct.h.

11.165.2.2 Payload [CFE_ES_SetMaxPRCountCmd_Payload_t](#) CFE_ES_SetMaxPRCountCmd::Payload
Command payload.

Definition at line 186 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.166 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference

Set Maximum Processor Reset Count Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [uint16 MaxPRCount](#)

New maximum number of Processor Resets before an automatic Power-On Reset is performed.

11.166.1 Detailed Description

Set Maximum Processor Reset Count Command Payload.

For command details, see [CFE_ES_SET_MAX_PR_COUNT_CC](#)

Definition at line 128 of file default_cfe_es_msgdefs.h.

11.166.2 Field Documentation

11.166.2.1 MaxPRCount [uint16](#) CFE_ES_SetMaxPRCountCmd_Payload::MaxPRCount

New maximum number of Processor Resets before an automatic Power-On Reset is performed.

Definition at line 130 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.167 CFE_ES_SetPerfFilterMaskCmd Struct Reference

Set Performance Analyzer Filter Mask Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

- [CFE_ES_SetPerfFilterMaskCmd_Payload_t](#) Payload

Command payload.

11.167.1 Detailed Description

Set Performance Analyzer Filter Mask Command.
Definition at line 219 of file default_cfe_es_msgstruct.h.

11.167.2 Field Documentation

11.167.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_SetPerfFilterMaskCmd::CommandHeader

Command header.

Definition at line 221 of file default_cfe_es_msgstruct.h.

11.167.2.2 Payload [CFE_ES_SetPerfFilterMaskCmd_Payload_t](#) CFE_ES_SetPerfFilterMaskCmd::Payload

Command payload.

Definition at line 222 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.168 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference

Set Performance Analyzer Filter Mask Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [uint32 FilterMaskNum](#)
Index into array of Filter Masks.
- [uint32 FilterMask](#)
New Mask for specified entry in array of Filter Masks.

11.168.1 Detailed Description

Set Performance Analyzer Filter Mask Command Payload.
For command details, see [CFE_ES_SET_PERF_FILTER_MASK_CC](#)
Definition at line 189 of file default_cfe_es_msgdefs.h.

11.168.2 Field Documentation

11.168.2.1 FilterMask [uint32](#) CFE_ES_SetPerfFilterMaskCmd_Payload::FilterMask

New Mask for specified entry in array of Filter Masks.

Definition at line 192 of file default_cfe_es_msgdefs.h.

11.168.2.2 FilterMaskNum [uint32](#) CFE_ES_SetPerfFilterMaskCmd_Payload::FilterMaskNum

Index into array of Filter Masks.

Definition at line 191 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.169 CFE_ES_SetPerfTriggerMaskCmd Struct Reference

Set Performance Analyzer Trigger Mask Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_SetPerfTrigMaskCmd_Payload_t](#) Payload
Command payload.

11.169.1 Detailed Description

Set Performance Analyzer Trigger Mask Command.

Definition at line 228 of file default_cfe_es_msgstruct.h.

11.169.2 Field Documentation

11.169.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_SetPerfTriggerMaskCmd::CommandHeader

Command header.

Definition at line 230 of file default_cfe_es_msgstruct.h.

11.169.2.2 Payload [CFE_ES_SetPerfTrigMaskCmd_Payload_t](#) CFE_ES_SetPerfTriggerMaskCmd::Payload

Command payload.

Definition at line 231 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.170 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference

Set Performance Analyzer Trigger Mask Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [uint32](#) TriggerMaskNum
Index into array of Trigger Masks.
- [uint32](#) TriggerMask
New Mask for specified entry in array of Trigger Masks.

11.170.1 Detailed Description

Set Performance Analyzer Trigger Mask Command Payload.

For command details, see [CFE_ES_SET_PERF_TRIGGER_MASK_CC](#)

Definition at line 201 of file default_cfe_es_msgdefs.h.

11.170.2 Field Documentation

11.170.2.1 TriggerMask `uint32` `CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMask`

New Mask for specified entry in array of Trigger Masks.

Definition at line 204 of file `default_cfe_es_msgdefs.h`.

11.170.2.2 TriggerMaskNum `uint32` `CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMaskNum`

Index into array of Trigger Masks.

Definition at line 203 of file `default_cfe_es_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgdefs.h`

11.171 CFE_ES_StartApp Struct Reference

Start Application Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_ES_StartAppCmd_Payload_t Payload`
Command payload.

11.171.1 Detailed Description

Start Application Command.

Definition at line 142 of file `default_cfe_es_msgstruct.h`.

11.171.2 Field Documentation

11.171.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_ES_StartApp::CommandHeader`

Command header.

Definition at line 144 of file `default_cfe_es_msgstruct.h`.

11.171.2.2 Payload `CFE_ES_StartAppCmd_Payload_t` `CFE_ES_StartApp::Payload`

Command payload.

Definition at line 145 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

11.172 CFE_ES_StartAppCmd_Payload Struct Reference

Start Application Command Payload.

```
#include <default_cfe_es_msgdefs.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.
- [CFE_ES_MemOffset_t](#) [StackSize](#)
Desired stack size for the new application.
- [CFE_ES_ExceptionAction_Enum_t](#) [ExceptionAction](#)
CFE_ES_ExceptionAction_RESTART_APP=On exception, restart Application, CFE_ES_ExceptionAction_PROC_RESTART=On exception, perform a Processor Reset
- [CFE_ES_TaskPriority_Atom_t](#) [Priority](#)
The new Applications runtime priority.

11.172.1 Detailed Description

Start Application Command Payload.

For command details, see [CFE_ES_START_APP_CC](#)

Definition at line 82 of file default_cfe_es_msgdefs.h.

11.172.2 Field Documentation

11.172.2.1 AppEntryPoint char CFE_ES_StartAppCmd_Payload::AppEntryPoint [CFE_MISSION_MAX_API_LEN]
Symbolic name of Application's entry point.

Definition at line 85 of file default_cfe_es_msgdefs.h.

11.172.2.2 AppFileName char CFE_ES_StartAppCmd_Payload::AppFileName [CFE_MISSION_MAX_PATH_LEN]

Full path and filename of Application's executable image.

Definition at line 86 of file default_cfe_es_msgdefs.h.

11.172.2.3 Application char CFE_ES_StartAppCmd_Payload::Application [CFE_MISSION_MAX_API_LEN]

Name of Application to be started.

Definition at line 84 of file default_cfe_es_msgdefs.h.

11.172.2.4 ExceptionAction [CFE_ES_ExceptionAction_Enum_t](#) CFE_ES_StartAppCmd_Payload::Exception↔

Action

CFE_ES_ExceptionAction_RESTART_APP=On exception, restart Application, CFE_ES_ExceptionAction_PROC_RESTART=On exception, perform a Processor Reset

Definition at line 91 of file default_cfe_es_msgdefs.h.

11.172.2.5 Priority [CFE_ES_TaskPriority_Atom_t](#) CFE_ES_StartAppCmd_Payload::Priority

The new Applications runtime priority.

Definition at line 95 of file default_cfe_es_msgdefs.h.

11.172.2.6 StackSize [CFE_ES_MemOffset_t](#) CFE_ES_StartAppCmd_Payload::StackSize

Desired stack size for the new application.

Definition at line 89 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.173 CFE_ES_StartPerfCmd_Payload Struct Reference

Start Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

Data Fields

- [CFE_ES_PerfMode_Enum_t](#) TriggerMode

Desired trigger position (Start, Center, End). Values defined by CFE_ES_PerfMode.

11.173.1 Detailed Description

Start Performance Analyzer Command Payload.

For command details, see [CFE_ES_START_PERF_DATA_CC](#)

Definition at line 165 of file default_cfe_es_msgdefs.h.

11.173.2 Field Documentation

11.173.2.1 TriggerMode

[CFE_ES_PerfMode_Enum_t](#) CFE_ES_StartPerfCmd_Payload::TriggerMode

Desired trigger position (Start, Center, End). Values defined by [CFE_ES_PerfMode](#).

Definition at line 168 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgdefs.h

11.174 CFE_ES_StartPerfDataCmd Struct Reference

Start Performance Analyzer Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

- [CFE_ES_StartPerfCmd_Payload_t](#) Payload

Command payload.

11.174.1 Detailed Description

Start Performance Analyzer Command.

Definition at line 201 of file default_cfe_es_msgstruct.h.

11.174.2 Field Documentation

11.174.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_StartPerfDataCmd::CommandHeader
Command header.
Definition at line 203 of file default_cfe_es_msgstruct.h.

11.174.2.2 Payload [CFE_ES_StartPerfCmd_Payload_t](#) CFE_ES_StartPerfDataCmd::Payload
Command payload.
Definition at line 204 of file default_cfe_es_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.175 CFE_ES_StopAppCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_AppNameCmd_Payload_t](#) Payload
Command payload.

11.175.1 Detailed Description

Definition at line 153 of file default_cfe_es_msgstruct.h.

11.175.2 Field Documentation

11.175.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_ES_StopAppCmd::CommandHeader
Command header.
Definition at line 155 of file default_cfe_es_msgstruct.h.

11.175.2.2 Payload [CFE_ES_AppNameCmd_Payload_t](#) CFE_ES_StopAppCmd::Payload
Command payload.
Definition at line 156 of file default_cfe_es_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.176 CFE_ES_StopPerfCmd_Payload Struct Reference

Stop Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgdefs.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.

11.176.1 Detailed Description

Stop Performance Analyzer Command Payload.

For command details, see [CFE_ES_STOP_PERF_DATA_CC](#)

Definition at line 177 of file default_cfe_es_msgdefs.h.

11.176.2 Field Documentation

11.176.2.1 DataFileName `char CFE_ES_StopPerfCmd_Payload::DataFileName[CFE_MISSION_MAX_PATH_LEN]`

ASCII text string of full path and filename of file Performance Analyzer data is to be written.

Definition at line 179 of file default_cfe_es_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgdefs.h](#)

11.177 CFE_ES_StopPerfDataCmd Struct Reference

Stop Performance Analyzer Command.

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_ES_StopPerfCmd_Payload_t Payload](#)
Command payload.

11.177.1 Detailed Description

Stop Performance Analyzer Command.

Definition at line 210 of file default_cfe_es_msgstruct.h.

11.177.2 Field Documentation

11.177.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_StopPerfDataCmd::CommandHeader`

Command header.

Definition at line 212 of file default_cfe_es_msgstruct.h.

11.177.2.2 Payload `CFE_ES_StopPerfCmd_Payload_t CFE_ES_StopPerfDataCmd::Payload`

Command payload.

Definition at line 213 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.178 CFE_ES_TaskInfo Struct Reference

Task Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

Data Fields

- **CFE_ES_TaskId_t TaskId**
Task Id.
- **uint32 ExecutionCounter**
Task Execution Counter.
- **char TaskName [CFE_MISSION_MAX_API_LEN]**
Task Name.
- **CFE_ES_AppId_t AppId**
Parent Application ID.
- **charAppName [CFE_MISSION_MAX_API_LEN]**
Parent Application Name.
- **CFE_ES_MemOffset_t StackSize**
- **CFE_ES_TaskPriority_Atom_t Priority**
- **uint8 Spare [2]**

11.178.1 Detailed Description

Task Information.

Structure that is used to provide information about a task. It is primarily used for the Query All Tasks ([CFE_ES_QUERY_ALL_TASKS_CC](#)) command.

Note

There is not currently a telemetry message directly containing this data structure, but it does define the format of the data file generated by the Query All Tasks command. Therefore it should be considered part of the overall telemetry interface.

Definition at line 499 of file default_cfe_es_extern_typedefs.h.

11.178.2 Field Documentation

11.178.2.1 AppId [CFE_ES_AppId_t](#) CFE_ES_TaskInfo::AppId

Parent Application ID.

Definition at line 504 of file default_cfe_es_extern_typedefs.h.

11.178.2.2 AppName [char](#) CFE_ES_TaskInfo::AppName[[CFE_MISSION_MAX_API_LEN](#)]

Parent Application Name.

Definition at line 505 of file default_cfe_es_extern_typedefs.h.

11.178.2.3 ExecutionCounter [uint32](#) CFE_ES_TaskInfo::ExecutionCounter

Task Execution Counter.

Definition at line 502 of file default_cfe_es_extern_typedefs.h.

11.178.2.4 Priority [CFE_ES_TaskPriority_Atom_t](#) CFE_ES_TaskInfo::Priority

Priority of task

Definition at line 507 of file default_cfe_es_extern_typedefs.h.

11.178.2.5 Spare `uint8` `CFE_ES_TaskInfo::Spare[2]`

Spare bytes for alignment

Definition at line 508 of file default_cfe_es_extern_typedefs.h.

11.178.2.6 StackSize `CFE_ES_MemOffset_t` `CFE_ES_TaskInfo::StackSize`

Size of task stack

Definition at line 506 of file default_cfe_es_extern_typedefs.h.

11.178.2.7 TaskId `CFE_ES_TaskId_t` `CFE_ES_TaskInfo::TaskId`

Task Id.

Definition at line 501 of file default_cfe_es_extern_typedefs.h.

11.178.2.8 TaskName `char` `CFE_ES_TaskInfo::TaskName[CFE_MISSION_MAX_API_LEN]`

Task Name.

Definition at line 503 of file default_cfe_es_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_extern_typedefs.h](#)

11.179 CFE_ES_WriteERLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

Command header.

- `CFE_ES_FileNameCmd_Payload_t Payload`

Command payload.

11.179.1 Detailed Description

Definition at line 124 of file default_cfe_es_msgstruct.h.

11.179.2 Field Documentation

11.179.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_ES_WriteERLogCmd::CommandHeader`

Command header.

Definition at line 126 of file default_cfe_es_msgstruct.h.

11.179.2.2 Payload `CFE_ES_FileNameCmd_Payload_t` `CFE_ES_WriteERLogCmd::Payload`

Command payload.

Definition at line 127 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default_cfe_es_msgstruct.h](#)

11.180 CFE_ES_WriteSysLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** CommandHeader
Command header.
- **CFE_ES_FileNameCmd_Payload_t** Payload
Command payload.

11.180.1 Detailed Description

Definition at line 118 of file default_cfe_es_msgstruct.h.

11.180.2 Field Documentation

11.180.2.1 CommandHeader **CFE_MSG_CommandHeader_t** CFE_ES_WriteSysLogCmd::CommandHeader

Command header.

Definition at line 120 of file default_cfe_es_msgstruct.h.

11.180.2.2 Payload **CFE_ES_FileNameCmd_Payload_t** CFE_ES_WriteSysLogCmd::Payload

Command payload.

Definition at line 121 of file default_cfe_es_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default_cfe_es_msgstruct.h

11.181 CFE_EVS_AddEventFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** CommandHeader
Command header.
- **CFE_EVS_AppNameEventIDMaskCmd_Payload_t** Payload
Command payload.

11.181.1 Detailed Description

Definition at line 203 of file default_cfe_evs_msgstruct.h.

11.181.2 Field Documentation

11.181.2.1 CommandHeader **CFE_MSG_CommandHeader_t** CFE_EVS_AddEventFilterCmd::CommandHeader

Command header.

Definition at line 205 of file default_cfe_evs_msgstruct.h.

11.181.2.2 Payload `CFE_EVS_AppNameEventIDMaskCmd_Payload_t` `CFE_EVS_AddEventFilterCmd::Payload`
Command payload.

Definition at line 206 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.182 CFE_EVS_AppDataCmd_Payload Struct Reference

Write Event Services Application Information to File Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- char `AppDataFilename` [`CFE_MISSION_MAX_PATH_LEN`]
Filename where application data is to be written.

11.182.1 Detailed Description

Write Event Services Application Information to File Command Payload.

For command details, see [CFE_EVS_WRITE_APP_DATA_FILE_CC](#)

Definition at line 68 of file default_cfe_evs_msgdefs.h.

11.182.2 Field Documentation

11.182.2.1 AppDataFilename `char CFE_EVS_AppDataCmd_Payload::AppDataFilename` [`CFE_MISSION_MAX_PATH_LEN`]
Filename where application data is to be written.

Definition at line 70 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.183 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference

Generic App Name and Bitmask Command Payload.

```
#include <default_cfe_evs_msgdefs.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.

11.183.1 Detailed Description

Generic App Name and Bitmask Command Payload.

For command details, see [CFE_EVS_ENABLE_APP_EVENT_TYPE_CC](#) and/or [CFE_EVS_DISABLE_APP_EVENT_TYPE_CC](#)

Definition at line 140 of file default_cfe_evs_msgdefs.h.

11.183.2 Field Documentation

11.183.2.1 **AppName** `char CFE_EVS_AppNameBitMaskCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

Definition at line 142 of file default_cfe_evs_msgdefs.h.

11.183.2.2 **BitMask** `uint8 CFE_EVS_AppNameBitMaskCmd_Payload::BitMask`

BitMask to use in the command.

Definition at line 143 of file default_cfe_evs_msgdefs.h.

11.183.2.3 **Spare** `uint8 CFE_EVS_AppNameBitMaskCmd_Payload::Spare`

Pad to even byte.

Definition at line 144 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.184 CFE_EVS_AppNameCmd_Payload Struct Reference

Generic App Name Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- `char AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

11.184.1 Detailed Description

Generic App Name Command Payload.

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 117 of file default_cfe_evs_msgdefs.h.

11.184.2 Field Documentation

11.184.2.1 **AppName** `char CFE_EVS_AppNameCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

Definition at line 119 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.185 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference

Generic App Name and Event ID Command Payload.

```
#include <default_cfe_evs_msgdefs.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.

11.185.1 Detailed Description

Generic App Name and Event ID Command Payload.

For command details, see [CFE_EVS_RESET_FILTER_CC](#) and [CFE_EVS_DELETE_EVENT_FILTER_CC](#)

Definition at line 128 of file default_cfe_evs_msgdefs.h.

11.185.2 Field Documentation

11.185.2.1 AppName [char CFE_EVS_AppNameEventIDCmd_Payload::AppName \[CFE_MISSION_MAX_API_LEN\]](#)

Application name to use in the command.

Definition at line 130 of file default_cfe_evs_msgdefs.h.

11.185.2.2 EventID [uint16 CFE_EVS_AppNameEventIDCmd_Payload::EventID](#)

Event ID to use in the command.

Definition at line 131 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.186 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference

Generic App Name, Event ID, Mask Command Payload.

#include <default_cfe_evs_msgdefs.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.

11.186.1 Detailed Description

Generic App Name, Event ID, Mask Command Payload.

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 154 of file default_cfe_evs_msgdefs.h.

11.186.2 Field Documentation

11.186.2.1 AppName `char CFE_EVS_AppNameEventIDMaskCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`
Application name to use in the command.
Definition at line 156 of file default_cfe_evs_msgdefs.h.

11.186.2.2 EventID `uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::EventID`
Event ID to use in the command.
Definition at line 157 of file default_cfe_evs_msgdefs.h.

11.186.2.3 Mask `uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::Mask`
Mask to use in the command.
Definition at line 158 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.187 CFE_EVS_AppTlmData Struct Reference

#include <default_cfe_evs_msgdefs.h>

Data Fields

- **CFE_ES_AppId_t AppID**
Numerical application identifier.
- **uint16 AppMessageSentCounter**
Application message sent counter.
- **uint8 AppEnableStatus**
Application event service enable status.
- **uint8 AppMessageSquelchedCounter**
Number of events squelched.

11.187.1 Detailed Description

Definition at line 167 of file default_cfe_evs_msgdefs.h.

11.187.2 Field Documentation

11.187.2.1 AppEnableStatus `uint8 CFE_EVS_AppTlmData::AppEnableStatus`
Application event service enable status.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPENASTAT

Definition at line 173 of file default_cfe_evs_msgdefs.h.

11.187.2.2 AppID `CFE_ES_AppId_t CFE_EVS_AppTlmData::AppID`
Numerical application identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPID

Definition at line 169 of file default_cfe_evs_msgdefs.h.

11.187.2.3 AppMessageSentCounter `uint16` `CFE_EVS_AppTlmData::AppMessageSentCounter`
Application message sent counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPMGSENTC`

Definition at line 171 of file `default_cfe_evs_msgdefs.h`.

11.187.2.4 AppMessageSquelchedCounter `uint8` `CFE_EVS_AppTlmData::AppMessageSquelchedCounter`
Number of events squelched.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].SQUELCHEDC`

Definition at line 175 of file `default_cfe_evs_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgdefs.h`

11.188 CFE_EVS_BinFilter Struct Reference

Event message filter definition structure.

```
#include <cfe_evs_api_typedefs.h>
```

Data Fields

- **uint16 EventID**
Numerical event identifier.
- **uint16 Mask**
Binary filter mask value.

11.188.1 Detailed Description

Event message filter definition structure.

Definition at line 60 of file `cfe_evs_api_typedefs.h`.

11.188.2 Field Documentation

11.188.2.1 EventID `uint16` `CFE_EVS_BinFilter::EventID`

Numerical event identifier.

Definition at line 62 of file `cfe_evs_api_typedefs.h`.

11.188.2.2 Mask `uint16` `CFE_EVS_BinFilter::Mask`

Binary filter mask value.

Definition at line 63 of file `cfe_evs_api_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h`

11.189 CFE_EVS_BitMaskCmd_Payload Struct Reference

Generic Bitmask Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- `uint8 BitMask`
BitMask to use in the command.
- `uint8 Spare`
Pad to even byte.

11.189.1 Detailed Description

Generic Bitmask Command Payload.

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 104 of file default_cfe_evs_msgdefs.h.

11.189.2 Field Documentation**11.189.2.1 BitMask `uint8 CFE_EVS_BitMaskCmd_Payload::BitMask`**

BitMask to use in the command.

Definition at line 106 of file default_cfe_evs_msgdefs.h.

11.189.2.2 Spare `uint8 CFE_EVS_BitMaskCmd_Payload::Spare`

Pad to even byte.

Definition at line 107 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/[default_cfe_evs_msgdefs.h](#)

11.190 CFE_EVS_ClearLogCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.190.1 Detailed Description

Definition at line 60 of file default_cfe_evs_msgstruct.h.

11.190.2 Field Documentation**11.190.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_EVS_ClearLogCmd::CommandHeader`**
Command header.

Definition at line 62 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/[default_cfe_evs_msgstruct.h](#)

11.191 CFE_EVS_DeleteEventFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** CommandHeader
Command header.
- **CFE_EVS_AppNameEventIDCmd_Payload_t** Payload
Command payload.

11.191.1 Detailed Description

Definition at line 175 of file default_cfe_evs_msgstruct.h.

11.191.2 Field Documentation

11.191.2.1 CommandHeader **CFE_MSG_CommandHeader_t** CFE_EVS_DeleteEventFilterCmd::CommandHeader

Command header.

Definition at line 177 of file default_cfe_evs_msgstruct.h.

11.191.2.2 Payload **CFE_EVS_AppNameEventIDCmd_Payload_t** CFE_EVS_DeleteEventFilterCmd::Payload

Command payload.

Definition at line 178 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.192 CFE_EVS_DisableAppEventsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** CommandHeader
Command header.
- **CFE_EVS_AppNameCmd_Payload_t** Payload
Command payload.

11.192.1 Detailed Description

Definition at line 146 of file default_cfe_evs_msgstruct.h.

11.192.2 Field Documentation

11.192.2.1 CommandHeader **CFE_MSG_CommandHeader_t** CFE_EVS_DisableAppEventsCmd::CommandHeader

Command header.

Definition at line 148 of file default_cfe_evs_msgstruct.h.

11.192.2.2 Payload `CFE_EVS_AppNameCmd_Payload_t` `CFE_EVS_DisableAppEventsCmd::Payload`
Command payload.

Definition at line 149 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.193 CFE_EVS_DisableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_AppNameBitMaskCmd_Payload_t Payload`
Command payload.

11.193.1 Detailed Description

Definition at line 192 of file default_cfe_evs_msgstruct.h.

11.193.2 Field Documentation

11.193.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_EVS_DisableEventTypeCmd::CommandHeader`
Command header.

Definition at line 194 of file default_cfe_evs_msgstruct.h.

11.193.2.2 Payload `CFE_EVS_AppNameBitMaskCmd_Payload_t` `CFE_EVS_DisableEventTypeCmd::Payload`
Command payload.

Definition at line 195 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.194 CFE_EVS_DisableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_BitMaskCmd_Payload_t Payload`
Command payload.

11.194.1 Detailed Description

Definition at line 129 of file default_cfe_evs_msgstruct.h.

11.194.2 Field Documentation

11.194.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_EVS_DisableEventTypeCmd::CommandHeader`
Command header.

Definition at line 131 of file `default_cfe_evs_msgstruct.h`.

11.194.2.2 Payload `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_DisableEventTypeCmd::Payload`
Command payload.

Definition at line 132 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.195 CFE_EVS_DisablePortsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_BitMaskCmd_Payload_t Payload`
Command payload.

11.195.1 Detailed Description

Definition at line 117 of file `default_cfe_evs_msgstruct.h`.

11.195.2 Field Documentation

11.195.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_EVS_DisablePortsCmd::CommandHeader`
Command header.

Definition at line 119 of file `default_cfe_evs_msgstruct.h`.

11.195.2.2 Payload `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_DisablePortsCmd::Payload`
Command payload.

Definition at line 120 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.196 CFE_EVS_EnableAppEventsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_AppNameCmd_Payload_t](#) Payload
Command payload.

11.196.1 Detailed Description

Definition at line 140 of file default_cfe_evs_msgstruct.h.

11.196.2 Field Documentation

11.196.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_EnableAppEventsCmd::CommandHeader

Command header.

Definition at line 142 of file default_cfe_evs_msgstruct.h.

11.196.2.2 Payload [CFE_EVS_AppNameCmd_Payload_t](#) CFE_EVS_EnableAppEventsCmd::Payload

Command payload.

Definition at line 143 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.197 CFE_EVS_EnableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_AppNameBitMaskCmd_Payload_t](#) Payload
Command payload.

11.197.1 Detailed Description

Definition at line 186 of file default_cfe_evs_msgstruct.h.

11.197.2 Field Documentation

11.197.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_EnableEventTypeCmd::CommandHeader

Command header.

Definition at line 188 of file default_cfe_evs_msgstruct.h.

11.197.2.2 Payload `CFE_EVS_AppNameBitMaskCmd_Payload_t` `CFE_EVS_EnableAppEventTypeCmd::Payload`
Command payload.

Definition at line 189 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.198 CFE_EVS_EnableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_BitMaskCmd_Payload_t Payload`
Command payload.

11.198.1 Detailed Description

Definition at line 123 of file default_cfe_evs_msgstruct.h.

11.198.2 Field Documentation

11.198.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_EVS_EnableEventTypeCmd::CommandHeader`
Command header.

Definition at line 125 of file default_cfe_evs_msgstruct.h.

11.198.2.2 Payload `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_EnableEventTypeCmd::Payload`
Command payload.

Definition at line 126 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.199 CFE_EVS_EnablePortsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_BitMaskCmd_Payload_t Payload`
Command payload.

11.199.1 Detailed Description

Definition at line 111 of file default_cfe_evs_msgstruct.h.

11.199.2 Field Documentation

11.199.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_EVS_EnablePortsCmd::CommandHeader`
Command header.

Definition at line 113 of file `default_cfe_evs_msgstruct.h`.

11.199.2.2 Payload `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_EnablePortsCmd::Payload`
Command payload.

Definition at line 114 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.200 CFE_EVS_HousekeepingTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_EVS_HousekeepingTlm_Payload_t Payload`
Telemetry payload.

11.200.1 Detailed Description

Definition at line 221 of file `default_cfe_evs_msgstruct.h`.

11.200.2 Field Documentation

11.200.2.1 Payload `CFE_EVS_HousekeepingTlm_Payload_t` `CFE_EVS_HousekeepingTlm::Payload`
Telemetry payload.

Definition at line 224 of file `default_cfe_evs_msgstruct.h`.

11.200.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_EVS_HousekeepingTlm::TelemetryHeader`
Telemetry header.

Definition at line 223 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.201 CFE_EVS_HousekeepingTlm_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.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.

11.201.1 Detailed Description

Name Event Services Housekeeping Telemetry Packet

Definition at line 182 of file default_cfe_evs_msgdefs.h.

11.201.2 Field Documentation

11.201.2.1 AppData `CFE_EVS_AppTlmData_t CFE_EVS_HousekeepingTlm_Payload::AppData[CFE_MISSION_ES_MAX_APPLICATIONS]`
Array of registered application table data.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS]

Definition at line 216 of file default_cfe_evs_msgdefs.h.

11.201.2.2 CommandCounter `uint8 CFE_EVS_HousekeepingTlm_Payload::CommandCounter`
EVS Command Counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_CMDPC`

Definition at line 184 of file default_cfe_evs_msgdefs.h.

11.201.2.3 CommandErrorCounter `uint8 CFE_EVS_HousekeepingTlm_Payload::CommandErrorCounter`
EVS Command Error Counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_CMDEC`

Definition at line 186 of file default_cfe_evs_msgdefs.h.

11.201.2.4 LogEnabled `uint8 CFE_EVS_HousekeepingTlm_Payload::LogEnabled`
Current event log enable/disable state.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_LOGENABLED`

Definition at line 207 of file default_cfe_evs_msgdefs.h.

11.201.2.5 LogFullFlag `uint8 CFE_EVS_HousekeepingTlm_Payload::LogFullFlag`
Local event log full flag.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_LOGFULL`

Definition at line 197 of file default_cfe_evs_msgdefs.h.

11.201.2.6 LogMode `uint8 CFE_EVS_HousekeepingTlm_Payload::LogMode`
Local event logging mode (overwrite/discard)

Telemetry Mnemonic(s) `$sc_$cpu_EVS_LOGMODE`

Definition at line 199 of file default_cfe_evs_msgdefs.h.

11.201.2.7 LogOverflowCounter `uint16 CFE_EVS_HousekeepingTlm_Payload::LogOverflowCounter`
Local event log overflow counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_LOGOVERFLOWC`

Definition at line 204 of file default_cfe_evs_msgdefs.h.

11.201.2.8 MessageFormatMode `uint8 CFE_EVS_HousekeepingTlm_Payload::MessageFormatMode`
Event message format mode (short/long)

Telemetry Mnemonic(s) `$sc_$cpu_EVS_MSGFMTMODE`

Definition at line 188 of file default_cfe_evs_msgdefs.h.

11.201.2.9 MessageSendCounter `uint16` `CFE_EVS_HousekeepingTlm_Payload::MessageSendCounter`
Event message send counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_MSGSENTC`

Definition at line 202 of file `default_cfe_evs_msgdefs.h`.

11.201.2.10 MessageTruncCounter `uint8` `CFE_EVS_HousekeepingTlm_Payload::MessageTruncCounter`
Event message truncation counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_MSGTRUNC`

Definition at line 190 of file `default_cfe_evs_msgdefs.h`.

11.201.2.11 OutputPort `uint8` `CFE_EVS_HousekeepingTlm_Payload::OutputPort`
Output port mask.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_OUTPUTPORT`

Definition at line 195 of file `default_cfe_evs_msgdefs.h`.

11.201.2.12 Spare1 `uint8` `CFE_EVS_HousekeepingTlm_Payload::Spare1`
Padding for 32 bit boundary.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_HK_SPARE1`

Definition at line 209 of file `default_cfe_evs_msgdefs.h`.

11.201.2.13 Spare2 `uint8` `CFE_EVS_HousekeepingTlm_Payload::Spare2`
Padding for 32 bit boundary.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_HK_SPARE2`

Definition at line 211 of file `default_cfe_evs_msgdefs.h`.

11.201.2.14 Spare3 `uint8` `CFE_EVS_HousekeepingTlm_Payload::Spare3`
Padding for 32 bit boundary.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_HK_SPARE3`

Definition at line 213 of file `default_cfe_evs_msgdefs.h`.

11.201.2.15 UnregisteredAppCounter `uint8` `CFE_EVS_HousekeepingTlm_Payload::UnregisteredAppCounter`
Unregistered application message send counter.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_UNREGAPPC`

Definition at line 193 of file `default_cfe_evs_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgdefs.h`

11.202 CFE_EVS_LogFileCmd_Payload Struct Reference

Write Event Log to File Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- char [LogFilename \[CFE_MISSION_MAX_PATH_LEN\]](#)

Filename where log data is to be written.

11.202.1 Detailed Description

Write Event Log to File Command Payload.

For command details, see [CFE_EVS_WRITE_LOG_DATA_FILE_CC](#)

Definition at line 57 of file default_cfe_evs_msgdefs.h.

11.202.2 Field Documentation

11.202.2.1 LogFilename `char CFE_EVS_LogFileCmd_Payload::LogFilename [CFE_MISSION_MAX_PATH_LEN]`

Filename where log data is to be written.

Definition at line 59 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.203 CFE_EVS_LongEventTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)

Telemetry header.

- [CFE_EVS_LongEventTlm_Payload_t Payload](#)

Telemetry payload.

11.203.1 Detailed Description

Definition at line 227 of file default_cfe_evs_msgstruct.h.

11.203.2 Field Documentation

11.203.2.1 Payload `CFE_EVS_LongEventTlm_Payload_t CFE_EVS_LongEventTlm::Payload`

Telemetry payload.

Definition at line 230 of file default_cfe_evs_msgstruct.h.

11.203.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_EVS_LongEventTlm::TelemetryHeader`
Telemetry header.

Definition at line 229 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

11.204 CFE_EVS_LongEventTlm_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.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.

11.204.1 Detailed Description

Name Event Message Telemetry Packet (Long format)

Definition at line 239 of file `default_cfe_evs_msgdefs.h`.

11.204.2 Field Documentation

11.204.2.1 Message `char CFE_EVS_LongEventTlm_Payload::Message[CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]`
Event message string.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_EVENT[CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]`

Definition at line 242 of file `default_cfe_evs_msgdefs.h`.

11.204.2.2 PacketID `CFE_EVS_PacketID_t CFE_EVS_LongEventTlm_Payload::PacketID`
Event packet information.

Definition at line 241 of file `default_cfe_evs_msgdefs.h`.

11.204.2.3 Spare1 `uint8 CFE_EVS_LongEventTlm_Payload::Spare1`
Structure padding.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_SPARE1`

Definition at line 244 of file `default_cfe_evs_msgdefs.h`.

11.204.2.4 Spare2 `uint8 CFE_EVS_LongEventTlm_Payload::Spare2`
Structure padding.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_SPARE2

Definition at line 246 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.205 CFE_EVS_NoopCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t CommandHeader**
Command header.

11.205.1 Detailed Description

Definition at line 50 of file default_cfe_evs_msgstruct.h.

11.205.2 Field Documentation

11.205.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_EVS_NoopCmd::CommandHeader`
Command header.

Definition at line 54 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.206 CFE_EVS_PacketID Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- `char AppName [CFE_MISSION_MAX_API_LEN]`
Application name.
- `uint16 EventID`
Numerical event identifier.
- `CFE_EVS_EventType_Enum_t EventType`
Numerical event type identifier.
- `uint32 SpacecraftID`
Spacecraft identifier.
- `uint32 ProcessorID`
Numerical processor identifier.

11.206.1 Detailed Description

Telemetry packet structures

Definition at line 222 of file default_cfe_evs_msgdefs.h.

11.206.2 Field Documentation

11.206.2.1 AppName `char CFE_EVS_PacketID::AppName[CFE_MISSION_MAX_API_LEN]`
Application name.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_APPNAME[OS_MAX_API_NAME]`

Definition at line 224 of file default_cfe_evs_msgdefs.h.

11.206.2.2 EventID `uint16 CFE_EVS_PacketID::EventID`
Numerical event identifier.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_EVENTID`

Definition at line 226 of file default_cfe_evs_msgdefs.h.

11.206.2.3 EventType `CFE_EVS_EventType_Enum_t CFE_EVS_PacketID::EventType`
Numerical event type identifier.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_EVENTTYPE`

Definition at line 228 of file default_cfe_evs_msgdefs.h.

11.206.2.4 ProcessorID `uint32 CFE_EVS_PacketID::ProcessorID`
Numerical processor identifier.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_PROCESSORID`

Definition at line 232 of file default_cfe_evs_msgdefs.h.

11.206.2.5 SpacecraftID `uint32 CFE_EVS_PacketID::SpacecraftID`
Spacecraft identifier.

Telemetry Mnemonic(s) `$sc_$cpu_EVS_SCID`

Definition at line 230 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.207 CFE_EVS_ResetAllFiltersCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_AppNameCmd_Payload_t Payload`
Command payload.

11.207.1 Detailed Description

Definition at line 158 of file default_cfe_evs_msgstruct.h.

11.207.2 Field Documentation

11.207.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE_EVS_ResetAllFiltersCmd::CommandHeader

Command header.

Definition at line 160 of file default_cfe_evs_msgstruct.h.

11.207.2.2 Payload `CFE_EVS_AppNameCmd_Payload_t` CFE_EVS_ResetAllFiltersCmd::Payload

Command payload.

Definition at line 161 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.208 CFE_EVS_ResetAppCounterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader
Command header.
- `CFE_EVS_AppNameCmd_Payload_t` Payload
Command payload.

11.208.1 Detailed Description

Definition at line 152 of file default_cfe_evs_msgstruct.h.

11.208.2 Field Documentation

11.208.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE_EVS_ResetAppCounterCmd::CommandHeader

Command header.

Definition at line 154 of file default_cfe_evs_msgstruct.h.

11.208.2.2 Payload `CFE_EVS_AppNameCmd_Payload_t` CFE_EVS_ResetAppCounterCmd::Payload

Command payload.

Definition at line 155 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.209 CFE_EVS_ResetCountersCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.209.1 Detailed Description

Definition at line 55 of file default_cfe_evs_msgstruct.h.

11.209.2 Field Documentation

11.209.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_ResetCountersCmd::CommandHeader

Command header.

Definition at line 57 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.210 CFE_EVS_ResetFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_AppNameEventIDCmd_Payload_t](#) Payload
Command payload.

11.210.1 Detailed Description

Definition at line 169 of file default_cfe_evs_msgstruct.h.

11.210.2 Field Documentation

11.210.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_ResetFilterCmd::CommandHeader

Command header.

Definition at line 171 of file default_cfe_evs_msgstruct.h.

11.210.2.2 Payload [CFE_EVS_AppNameEventIDCmd_Payload_t](#) CFE_EVS_ResetFilterCmd::Payload

Command payload.

Definition at line 172 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.211 CFE_EVS_SendHkCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

11.211.1 Detailed Description

Definition at line 65 of file default_cfe_evs_msgstruct.h.

11.211.2 Field Documentation**11.211.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_SendHkCmd::CommandHeader**

Command header.

Definition at line 67 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.212 CFE_EVS_SetEventFormatCode_Payload Struct Reference

Set Event Format Mode Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- [CFE_EVS_MsgFormat_Enum_t](#) MsgFormat

Mode to use in the command.

- [uint8](#) Spare

Pad to even byte.

11.212.1 Detailed Description

Set Event Format Mode Command Payload.

For command details, see [CFE_EVS_SET_EVENT_FORMAT_MODE_CC](#)

Definition at line 91 of file default_cfe_evs_msgdefs.h.

11.212.2 Field Documentation**11.212.2.1 MsgFormat [CFE_EVS_MsgFormat_Enum_t](#) CFE_EVS_SetEventFormatCode_Payload::MsgFormat**

Mode to use in the command.

Definition at line 93 of file default_cfe_evs_msgdefs.h.

11.212.2.2 Spare [uint8](#) CFE_EVS_SetEventFormatCode_Payload::Spare

Pad to even byte.

Definition at line 94 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.213 CFE_EVS_SetEventFormatModeCmd Struct Reference

Set Event Format Mode Command.

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_SetEventFormatMode_Payload_t](#) Payload
Command payload.

11.213.1 Detailed Description

Set Event Format Mode Command.

Definition at line 100 of file default_cfe_evs_msgstruct.h.

11.213.2 Field Documentation

11.213.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_SetEventFormatModeCmd::CommandHeader
Command header.

Definition at line 102 of file default_cfe_evs_msgstruct.h.

11.213.2.2 Payload [CFE_EVS_SetEventFormatMode_Payload_t](#) CFE_EVS_SetEventFormatModeCmd::Payload
Command payload.

Definition at line 103 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.214 CFE_EVS_SetFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_AppNameEventIDMaskCmd_Payload_t](#) Payload
Command payload.

11.214.1 Detailed Description

Definition at line 209 of file default_cfe_evs_msgstruct.h.

11.214.2 Field Documentation

11.214.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_SetFilterCmd::CommandHeader
Command header.
Definition at line 211 of file default_cfe_evs_msgstruct.h.

11.214.2.2 Payload [CFE_EVS_AppNameEventIDMaskCmd_Payload_t](#) CFE_EVS_SetFilterCmd::Payload
Command payload.
Definition at line 212 of file default_cfe_evs_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.215 CFE_EVS_SetLogMode_Payload Struct Reference

Set Log Mode Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- [CFE_EVS_LogMode_Enum_t LogMode](#)
Mode to use in the command.
- [uint8 Spare](#)
Pad to even byte.

11.215.1 Detailed Description

Set Log Mode Command Payload.

For command details, see [CFE_EVS_SET_LOG_MODE_CC](#)

Definition at line 79 of file default_cfe_evs_msgdefs.h.

11.215.2 Field Documentation

11.215.2.1 LogMode [CFE_EVS_LogMode_Enum_t](#) CFE_EVS_SetLogMode_Payload::LogMode
Mode to use in the command.
Definition at line 81 of file default_cfe_evs_msgdefs.h.

11.215.2.2 Spare [uint8](#) CFE_EVS_SetLogMode_Payload::Spare
Pad to even byte.
Definition at line 82 of file default_cfe_evs_msgdefs.h.
The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.216 CFE_EVS_SetLogModeCmd Struct Reference

Set Log Mode Command.

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_SetLogMode_Payload_t](#) Payload
Command payload.

11.216.1 Detailed Description

Set Log Mode Command.

Definition at line 91 of file default_cfe_evs_msgstruct.h.

11.216.2 Field Documentation

11.216.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_SetLogModeCmd::CommandHeader

Command header.

Definition at line 93 of file default_cfe_evs_msgstruct.h.

11.216.2.2 Payload [CFE_EVS_SetLogMode_Payload_t](#) CFE_EVS_SetLogModeCmd::Payload

Command payload.

Definition at line 94 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.217 CFE_EVS_ShortEventTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_EVS_ShortEventTlm_Payload_t](#) Payload
Telemetry payload.

11.217.1 Detailed Description

Definition at line 233 of file default_cfe_evs_msgstruct.h.

11.217.2 Field Documentation

11.217.2.1 Payload [CFE_EVS_ShortEventTlm_Payload_t](#) CFE_EVS_ShortEventTlm::Payload

Telemetry payload.

Definition at line 236 of file default_cfe_evs_msgstruct.h.

11.217.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CFE_EVS_ShortEventTlm::TelemetryHeader
Telemetry header.

Definition at line 235 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.218 CFE_EVS_ShortEventTlm_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

Data Fields

- [CFE_EVS_PacketID_t](#) PacketID

Event packet information.

11.218.1 Detailed Description

Name Event Message Telemetry Packet (Short format)

Definition at line 253 of file default_cfe_evs_msgdefs.h.

11.218.2 Field Documentation

11.218.2.1 PacketID [CFE_EVS_PacketID_t](#) CFE_EVS_ShortEventTlm_Payload::PacketID

Event packet information.

Definition at line 255 of file default_cfe_evs_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgdefs.h

11.219 CFE_EVS_WriteAppDataFileCmd Struct Reference

Write Event Services Application Information to File Command.

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

- [CFE_EVS_AppDataCmd_Payload_t](#) Payload

Command payload.

11.219.1 Detailed Description

Write Event Services Application Information to File Command.

Definition at line 82 of file default_cfe_evs_msgstruct.h.

11.219.2 Field Documentation

11.219.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_WriteAppDataFileCmd::CommandHeader
Command header.

Definition at line 84 of file default_cfe_evs_msgstruct.h.

11.219.2.2 Payload [CFE_EVS_AppDataCmd_Payload_t](#) CFE_EVS_WriteAppDataFileCmd::Payload
Command payload.

Definition at line 85 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.220 CFE_EVS_WriteLogFileCmd Struct Reference

Write Event Log to File Command.

```
#include <default_cfe_evs_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_LogFileCmd_Payload_t](#) Payload
Command payload.

11.220.1 Detailed Description

Write Event Log to File Command.

Definition at line 73 of file default_cfe_evs_msgstruct.h.

11.220.2 Field Documentation

11.220.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_EVS_WriteLogFileCmd::CommandHeader
Command header.

Definition at line 75 of file default_cfe_evs_msgstruct.h.

11.220.2.2 Payload [CFE_EVS_LogFileCmd_Payload_t](#) CFE_EVS_WriteLogFileCmd::Payload
Command payload.

Definition at line 76 of file default_cfe_evs_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default_cfe_evs_msgstruct.h

11.221 CFE_FS_FileWriteMetaData Struct Reference

External Metadata/State object associated with background file writes.

```
#include <cfe_fs_api_typedefs.h>
```

Data Fields

- volatile bool [IsPending](#)
- char [FileName](#) [[CFE_MISSION_MAX_PATH_LEN](#)]
- [uint32 FileSubType](#)
- char [Description](#) [[CFE_FS_HDR_DESC_MAX_LEN](#)]
- [CFE_FS_FileWriteGetData_t GetData](#)
- [CFE_FS_FileWriteOnEvent_t OnEvent](#)

11.221.1 Detailed Description

External Metadata/State object associated with background file writes.

Applications intending to schedule background file write jobs should instantiate this object in static/global data memory. This keeps track of the state of the file write request(s).

Definition at line 124 of file [cfe_fs_api_typedefs.h](#).

11.221.2 Field Documentation

11.221.2.1 **Description** [char CFE_FS_FileWriteMetaData::Description\[CFE_FS_HDR_DESC_MAX_LEN\]](#)

Description of file (for FS header)

Definition at line 132 of file [cfe_fs_api_typedefs.h](#).

11.221.2.2 **FileName** [char CFE_FS_FileWriteMetaData::FileName\[CFE_MISSION_MAX_PATH_LEN\]](#)

Name of file to write

Definition at line 128 of file [cfe_fs_api_typedefs.h](#).

11.221.2.3 **FileSubType** [uint32 CFE_FS_FileWriteMetaData::FileSubType](#)

Type of file to write (for FS header)

Definition at line 131 of file [cfe_fs_api_typedefs.h](#).

11.221.2.4 **GetData** [CFE_FS_FileWriteGetData_t CFE_FS_FileWriteMetaData::GetData](#)

Application callback to get a data record

Definition at line 134 of file [cfe_fs_api_typedefs.h](#).

11.221.2.5 **IsPending** [volatile bool CFE_FS_FileWriteMetaData::IsPending](#)

Whether request is pending (volatile as it may be checked outside lock)

Definition at line 126 of file [cfe_fs_api_typedefs.h](#).

11.221.2.6 **OnEvent** [CFE_FS_FileWriteOnEvent_t CFE_FS_FileWriteMetaData::OnEvent](#)

Application callback for abstract event processing

Definition at line 135 of file [cfe_fs_api_typedefs.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h](#)

11.222 CFE_FS_Header Struct Reference

Standard cFE File header structure definition.

```
#include <default_cfe_fs_filedef.h>
```

Data Fields

- **uint32 ContentType**
Identifies the content type ('cFE1'=0x63464531)
- **uint32 SubType**
Type of ContentType, if necessary.
- **uint32 Length**
Length of this header to support external processing.
- **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.

11.222.1 Detailed Description

Standard cFE File header structure definition.

Definition at line 181 of file default_cfe_fs_filedef.h.

11.222.2 Field Documentation

11.222.2.1 ApplicationID `uint32 CFE_FS_Header::ApplicationID`

Application that generated the file.

Definition at line 190 of file default_cfe_fs_filedef.h.

11.222.2.2 ContentType `uint32 CFE_FS_Header::ContentType`

Identifies the content type ('cFE1'=0x63464531)

Definition at line 183 of file default_cfe_fs_filedef.h.

11.222.2.3 Description `char CFE_FS_Header::Description[CFE_FS_HDR_DESC_MAX_LEN]`

File description.

Definition at line 195 of file default_cfe_fs_filedef.h.

11.222.2.4 Length `uint32 CFE_FS_Header::Length`

Length of this header to support external processing.

Definition at line 187 of file `default_cfe_fs_filedef.h`.

11.222.2.5 ProcessorID `uint32 CFE_FS_Header::ProcessorID`

Processor that generated the file.

Definition at line 189 of file `default_cfe_fs_filedef.h`.

11.222.2.6 SpacecraftID `uint32 CFE_FS_Header::SpacecraftID`

Spacecraft that generated the file.

Definition at line 188 of file `default_cfe_fs_filedef.h`.

11.222.2.7 SubType `uint32 CFE_FS_Header::SubType`

Type of ContentType, if necessary.

Standard SubType definitions can be found [here](#)

Definition at line 184 of file `default_cfe_fs_filedef.h`.

11.222.2.8 TimeSeconds `uint32 CFE_FS_Header::TimeSeconds`

File creation timestamp (seconds)

Definition at line 192 of file `default_cfe_fs_filedef.h`.

11.222.2.9 TimeSubSeconds `uint32 CFE_FS_Header::TimeSubSeconds`

File creation timestamp (sub-seconds)

Definition at line 193 of file `default_cfe_fs_filedef.h`.

The documentation for this struct was generated from the following file:

- cfe/modules/fs/config/[default_cfe_fs_filedef.h](#)

11.223 CFE_SB_AllSubscriptionsTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- **CFE_MSG_TelemetryHeader_t TelemetryHeader**
Telemetry header.
- **CFE_SB_AllSubscriptionsTlm_Payload_t Payload**
Telemetry payload.

11.223.1 Detailed Description

Definition at line 145 of file `default_cfe_sb_msgstruct.h`.

11.223.2 Field Documentation

11.223.2.1 Payload `CFE_SB_AllSubscriptionsTlm_Payload_t` `CFE_SB_AllSubscriptionsTlm::Payload`
Telemetry payload.

Definition at line 148 of file `default_cfe_sb_msgstruct.h`.

11.223.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_SB_AllSubscriptionsTlm::TelemetryHeader`
Telemetry header.

Definition at line 147 of file `default_cfe_sb_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

11.224 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.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.

11.224.1 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 274 of file `default_cfe_sb_msgdefs.h`.

11.224.2 Field Documentation

11.224.2.1 Entries `uint32 CFE_SB_AllSubscriptionsTlm_Payload::Entries`

Number of entries in the pkt.

Definition at line 278 of file `default_cfe_sb_msgdefs.h`.

11.224.2.2 Entry `CFE_SB_SubEntries_t CFE_SB_AllSubscriptionsTlm_Payload::Entry [CFE_SB_SUB_ENTRIES_PER_PKT]`

Array of `CFE_SB_SubEntries_t` entries.

Definition at line 279 of file `default_cfe_sb_msgdefs.h`.

11.224.2.3 PktSegment `uint32 CFE_SB_AllSubscriptionsTlm_Payload::PktSegment`
Pkt number(starts at 1) in the series.

Definition at line 276 of file default_cfe_sb_msgdefs.h.

11.224.2.4 TotalSegments `uint32 CFE_SB_AllSubscriptionsTlm_Payload::TotalSegments`

Total number of pkts needed to complete the request.

Definition at line 277 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgdefs.h](#)

11.225 CFE_SB_DisableRouteCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_SB_RouteCmd_Payload_t Payload](#)
Command payload.

11.225.1 Detailed Description

Definition at line 117 of file default_cfe_sb_msgstruct.h.

11.225.2 Field Documentation

11.225.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_SB_DisableRouteCmd::CommandHeader`
Command header.

Definition at line 119 of file default_cfe_sb_msgstruct.h.

11.225.2.2 Payload `CFE_SB_RouteCmd_Payload_t CFE_SB_DisableRouteCmd::Payload`
Command payload.

Definition at line 120 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgstruct.h](#)

11.226 CFE_SB_DisableSubReportingCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)

11.226.1 Detailed Description

Definition at line 67 of file default_cfe_sb_msgstruct.h.

11.226.2 Field Documentation

11.226.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_SB_DisableSubReportingCmd::CommandHeader

Definition at line 69 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.227 CFE_SB_EnableRouteCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_SB_RouteCmd_Payload_t](#) Payload
Command payload.

11.227.1 Detailed Description

Definition at line 111 of file default_cfe_sb_msgstruct.h.

11.227.2 Field Documentation

11.227.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_SB_EnableRouteCmd::CommandHeader

Command header.

Definition at line 113 of file default_cfe_sb_msgstruct.h.

11.227.2.2 Payload [CFE_SB_RouteCmd_Payload_t](#) CFE_SB_EnableRouteCmd::Payload

Command payload.

Definition at line 114 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.228 CFE_SB_EnableSubReportingCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

11.228.1 Detailed Description

Definition at line 62 of file default_cfe_sb_msgstruct.h.

11.228.2 Field Documentation

11.228.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_SB_EnableSubReportingCmd::CommandHeader`
Definition at line 64 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgstruct.h](#)

11.229 CFE_SB_HousekeepingTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_SB_HousekeepingTlm_Payload_t Payload`
Telemetry payload.

11.229.1 Detailed Description

Definition at line 127 of file default_cfe_sb_msgstruct.h.

11.229.2 Field Documentation

11.229.2.1 Payload `CFE_SB_HousekeepingTlm_Payload_t` `CFE_SB_HousekeepingTlm::Payload`
Telemetry payload.

Definition at line 132 of file default_cfe_sb_msgstruct.h.

11.229.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_SB_HousekeepingTlm::TelemetryHeader`
Telemetry header.

Definition at line 131 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgstruct.h](#)

11.230 CFE_SB_HousekeepingTlm_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.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`

- `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 GetPipeldByNameErrorCounter`
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

11.230.1 Detailed Description

Name Software Bus task housekeeping Packet

Definition at line 67 of file default_cfe_sb_msgdefs.h.

11.230.2 Field Documentation

11.230.2.1 CommandCounter `uint8 CFE_SB_HousekeepingTlm_Payload::CommandCounter`
Count of valid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_CMDPC

Definition at line 71 of file default_cfe_sb_msgdefs.h.

11.230.2.2 CommandErrorCounter `uint8 CFE_SB_HousekeepingTlm_Payload::CommandErrorCounter`
Count of invalid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_CMDEC

Definition at line 73 of file default_cfe_sb_msgdefs.h.

11.230.2.3 CreatePipeErrorCounter `uint8 CFE_SB_HousekeepingTlm_Payload::CreatePipeErrorCounter`
Count of errors in create pipe API.

Telemetry Mnemonic(s) `$sc_$cpu_SB_NewPipeEC`

Definition at line 84 of file default_cfe_sb_msgdefs.h.

11.230.2.4 DuplicateSubscriptionsCounter `uint8 CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptionsCounter`
Count of duplicate subscriptions.

Telemetry Mnemonic(s) `$sc_$cpu_SB_DupSubCnt`

Definition at line 90 of file default_cfe_sb_msgdefs.h.

11.230.2.5 GetPipeIdByNameErrorCounter `uint8 CFE_SB_HousekeepingTlm_Payload::GetPipeIdByNameErrorCounter`
Count of errors in get pipe id by name API.

Telemetry Mnemonic(s) `$sc_$cpu_SB_GetPipeIDByNameEC`

Definition at line 92 of file default_cfe_sb_msgdefs.h.

11.230.2.6 InternalErrorCounter `uint8 CFE_SB_HousekeepingTlm_Payload::InternalErrorCounter`
Count of queue read or write errors.

Telemetry Mnemonic(s) `$sc_$cpu_SB_InternalEC`

Definition at line 82 of file default_cfe_sb_msgdefs.h.

11.230.2.7 MemInUse `uint32 CFE_SB_HousekeepingTlm_Payload::MemInUse`
Memory in use.

Telemetry Mnemonic(s) `$sc_$cpu_SB_MemInUse`

Definition at line 105 of file default_cfe_sb_msgdefs.h.

11.230.2.8 MemPoolHandle `CFE_ES_MemHandle_t CFE_SB_HousekeepingTlm_Payload::MemPoolHandle`
Handle to SB's Memory Pool.

Telemetry Mnemonic(s) `$sc_$cpu_SB_MemPoolHdl`

Definition at line 102 of file default_cfe_sb_msgdefs.h.

11.230.2.9 MsgLimitErrorCounter `uint16 CFE_SB_HousekeepingTlm_Payload::MsgLimitErrorCounter`
Count of msg id to pipe errors.

Telemetry Mnemonic(s) `$sc_$cpu_SB_MsgLimEC`

Definition at line 99 of file default_cfe_sb_msgdefs.h.

11.230.2.10 MsgReceiveErrorCounter `uint8` `CFE_SB_HousekeepingTlm_Payload::MsgReceiveErrorCounter`
Count of message receive errors.

Telemetry Mnemonic(s) `$sc_$cpu_SB_MsgRecEC`

Definition at line 80 of file `default_cfe_sb_msgdefs.h`.

11.230.2.11 MsgSendErrorCounter `uint8` `CFE_SB_HousekeepingTlm_Payload::MsgSendErrorCounter`
Count of message send errors.

Telemetry Mnemonic(s) `$sc_$cpu_SB_MsgSndEC`

Definition at line 77 of file `default_cfe_sb_msgdefs.h`.

11.230.2.12 NoSubscribersCounter `uint8` `CFE_SB_HousekeepingTlm_Payload::NoSubscribersCounter`
Count pkts sent with no subscribers.

Telemetry Mnemonic(s) `$sc_$cpu_SB_NoSubEC`

Definition at line 75 of file `default_cfe_sb_msgdefs.h`.

11.230.2.13 PipeOptsErrorCounter `uint8` `CFE_SB_HousekeepingTlm_Payload::PipeOptsErrorCounter`
Count of errors in set/get pipe options API.

Telemetry Mnemonic(s) `$sc_$cpu_SB_PipeOptsEC`

Definition at line 88 of file `default_cfe_sb_msgdefs.h`.

11.230.2.14 PipeOverflowErrorCounter `uint16` `CFE_SB_HousekeepingTlm_Payload::PipeOverflowError←`
Counter
Count of pipe overflow errors.

Telemetry Mnemonic(s) `$sc_$cpu_SB_PipeOvrEC`

Definition at line 97 of file `default_cfe_sb_msgdefs.h`.

11.230.2.15 Spare2Align `uint8` `CFE_SB_HousekeepingTlm_Payload::Spare2Align[1]`
Spare bytes to ensure alignment.

Telemetry Mnemonic(s) `$sc_$cpu_SB_Spare2Align[2]`

Definition at line 94 of file `default_cfe_sb_msgdefs.h`.

11.230.2.16 SubscribeErrorCounter `uint8` `CFE_SB_HousekeepingTlm_Payload::SubscribeErrorCounter`
Count of errors in subscribe API.

Telemetry Mnemonic(s) `$sc_$cpu_SB_SubscrEC`

Definition at line 86 of file `default_cfe_sb_msgdefs.h`.

11.230.2.17 UnmarkedMem `uint32 CFE_SB_HousekeepingTlm_Payload::UnmarkedMem`
cfg param CFE_PLATFORM_SB_BUFSIZE minus Peak Memory in use

Telemetry Mnemonic(s) \$sc_\$cpu_SB_UnMarkedMem

Definition at line 108 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.231 CFE_SB_Msg Union Reference

Software Bus generic message.

```
#include <cfesb_api_typedefs.h>
```

Data Fields

- **CFE_MSG_Message_t** `Msg`
Base message type without enforced alignment.
- long long int **LongInt**
Align to support Long Integer.
- long double **LongDouble**
Align to support Long Double.

11.231.1 Detailed Description

Software Bus generic message.

Definition at line 142 of file cfe_sb_api_typedefs.h.

11.231.2 Field Documentation

11.231.2.1 LongDouble `long double CFE_SB_Msg::LongDouble`
Align to support Long Double.
Definition at line 146 of file cfe_sb_api_typedefs.h.

11.231.2.2 LongInt `long long int CFE_SB_Msg::LongInt`
Align to support Long Integer.
Definition at line 145 of file cfe_sb_api_typedefs.h.

11.231.2.3 Msg `CFE_MSG_Message_t CFE_SB_Msg::Msg`
Base message type without enforced alignment.
Definition at line 144 of file cfe_sb_api_typedefs.h.
Referenced by CF_AppPipe(), CF_CFDP_MsgOutGet(), CF_CFDP_ReceiveMessage(), CF_CFDP_Send(), and CF←_ProcessGroundCommand().
The documentation for this union was generated from the following file:

- cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h

11.232 CFE_SB_MsgId_t Struct Reference

[CFE_SB_MsgId_t](#) type definition.

```
#include <default_cfe_sb_extern_typedefs.h>
```

Data Fields

- [CFE_SB_MsgId_Atom_t Value](#)

11.232.1 Detailed Description

[CFE_SB_MsgId_t](#) type definition.

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 104 of file default_cfe_sb_extern_typedefs.h.

11.232.2 Field Documentation

11.232.2.1 Value [CFE_SB_MsgId_Atom_t](#) CFE_SB_MsgId_t::Value

Definition at line 106 of file default_cfe_sb_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h

11.233 CFE_SB_MsgMapFileEntry Struct Reference

SB Map File Entry.

```
#include <default_cfe_sb_msgdefs.h>
```

Data Fields

- [CFE_SB_MsgId_t MsgId](#)

Message Id which has been subscribed to.

- [CFE_SB_RoutId_Atom_t Index](#)

Routing raw index value (0 based, not Route ID)

11.233.1 Detailed Description

SB Map File Entry.

Structure of one element of the map information in response to [CFE_SB_WRITE_MAP_INFO_CC](#)

Definition at line 226 of file default_cfe_sb_msgdefs.h.

11.233.2 Field Documentation

11.233.2.1 Index `CFE_SB_RouteId_Atom_t` `CFE_SB_MsgMapFileEntry::Index`
Routing raw index value (0 based, not Route ID)
Definition at line 229 of file default_cfe_sb_msgdefs.h.

11.233.2.2 MsgId `CFE_SB_MsgId_t` `CFE_SB_MsgMapFileEntry::MsgId`
Message Id which has been subscribed to.
Definition at line 228 of file default_cfe_sb_msgdefs.h.
The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgdefs.h`

11.234 CFE_SB_NoopCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

11.234.1 Detailed Description

Definition at line 52 of file default_cfe_sb_msgstruct.h.

11.234.2 Field Documentation

11.234.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_SB_NoopCmd::CommandHeader`
Definition at line 58 of file default_cfe_sb_msgstruct.h.
The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

11.235 CFE_SB_PipeDepthStats Struct Reference

SB Pipe Depth Statistics.

```
#include <default_cfe_sb_msgdefs.h>
```

Data Fields

- `CFE_SB_Pipeld_t Pipeld`
Pipe Id associated with the stats below.
- `uint16 MaxQueueDepth`
Number of messages the pipe can hold.
- `uint16 CurrentQueueDepth`
Number of messages currently on the pipe.
- `uint16 PeakQueueDepth`
Peak number of messages that have been on the pipe.
- `uint16 Spare`
Spare word to ensure alignment.

11.235.1 Detailed Description

SB Pipe Depth Statistics.

Used in SB Statistics Telemetry Packet [CFE_SB_StatsTlm_t](#)

Definition at line 115 of file default_cfe_sb_msgdefs.h.

11.235.2 Field Documentation

11.235.2.1 CurrentQueueDepth [uint16](#) CFE_SB_PipeDepthStats::CurrentQueueDepth

Number of messages currently on the pipe.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDINUSE

Definition at line 121 of file default_cfe_sb_msgdefs.h.

11.235.2.2 MaxQueueDepth [uint16](#) CFE_SB_PipeDepthStats::MaxQueueDepth

Number of messages the pipe can hold.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDDEPTH

Definition at line 119 of file default_cfe_sb_msgdefs.h.

11.235.2.3 PeakQueueDepth [uint16](#) CFE_SB_PipeDepthStats::PeakQueueDepth

Peak number of messages that have been on the pipe.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDPKINUSE

Definition at line 123 of file default_cfe_sb_msgdefs.h.

11.235.2.4 PipeId [CFE_SB_PipeId_t](#) CFE_SB_PipeDepthStats::PipeId

Pipe Id associated with the stats below.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDPipeID

Definition at line 117 of file default_cfe_sb_msgdefs.h.

11.235.2.5 Spare [uint16](#) CFE_SB_PipeDepthStats::Spare

Spare word to ensure alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDSpare

Definition at line 125 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.236 CFE_SB_PipeInfoEntry Struct Reference

SB Pipe Information File Entry.

```
#include <default_cfe_sb_msgdefs.h>
```

Data Fields

- `CFE_SB_Pipeld_t Pipeld`
- `CFE_ES_AppId_t AppId`
- `char PipeName [CFE_MISSION_MAX_API_LEN]`
- `charAppName [CFE_MISSION_MAX_API_LEN]`
- `uint16 MaxQueueDepth`
- `uint16 CurrentQueueDepth`
- `uint16 PeakQueueDepth`
- `uint16 SendErrors`
- `uint8 Opts`
- `uint8 Spare [3]`

11.236.1 Detailed Description

SB Pipe Information File Entry.

This statistics structure is output as part of the CFE SB "Send Pipe Info" command (CFE_SB_SEND_PIPE_INFO_CC). Previous versions of CFE simply wrote the internal CFE_SB_PipeD_t object to the file, but this also contains information such as pointers which are not relevant outside the running CFE process.

By defining the pipe info structure separately, it also provides some independence, such that the internal CFE_SB_PipeD_t definition can evolve without changing the binary format of the information file.

Definition at line 144 of file default_cfe_sb_msgdefs.h.

11.236.2 Field Documentation

11.236.2.1 **AppId** `CFE_ES_AppId_t CFE_SB_PipeInfoEntry::AppId`

The runtime ID of the application that owns the pipe

Definition at line 147 of file default_cfe_sb_msgdefs.h.

11.236.2.2 **AppName** `char CFE_SB_PipeInfoEntry::AppName [CFE_MISSION_MAX_API_LEN]`

The Name of the application that owns the pipe

Definition at line 149 of file default_cfe_sb_msgdefs.h.

11.236.2.3 **CurrentQueueDepth** `uint16 CFE_SB_PipeInfoEntry::CurrentQueueDepth`

The current depth of the pipe

Definition at line 151 of file default_cfe_sb_msgdefs.h.

11.236.2.4 **MaxQueueDepth** `uint16 CFE_SB_PipeInfoEntry::MaxQueueDepth`

The allocated depth of the pipe (max capacity)

Definition at line 150 of file default_cfe_sb_msgdefs.h.

11.236.2.5 **Opts** `uint8 CFE_SB_PipeInfoEntry::Opts`

Pipe options set (bitmask)

Definition at line 154 of file default_cfe_sb_msgdefs.h.

11.236.2.6 PeakQueueDepth `uint16 CFE_SB_PipeInfoEntry::PeakQueueDepth`

The peak depth of the pipe (high watermark)

Definition at line 152 of file default_cfe_sb_msgdefs.h.

11.236.2.7 PipeId `CFE_SB_PipeId_t CFE_SB_PipeInfoEntry::PipeId`

The runtime ID of the pipe

Definition at line 146 of file default_cfe_sb_msgdefs.h.

11.236.2.8 PipeName `char CFE_SB_PipeInfoEntry::PipeName [CFE_MISSION_MAX_API_LEN]`

The Name of the pipe

Definition at line 148 of file default_cfe_sb_msgdefs.h.

11.236.2.9 SendErrors `uint16 CFE_SB_PipeInfoEntry::SendErrors`

Number of errors when writing to this pipe

Definition at line 153 of file default_cfe_sb_msgdefs.h.

11.236.2.10 Spare `uint8 CFE_SB_PipeInfoEntry::Spare[3]`

Padding to make this structure a multiple of 4 bytes

Definition at line 155 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.237 CFE_SB_Qos_t Struct Reference

Quality Of Service Type Definition.

```
#include <default_cfe_sb_extern_typedefs.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.

11.237.1 Detailed Description

Quality Of Service Type Definition.

Currently an unused parameter in `CFE_SB_SubscribeEx` Intended to be used for interprocessor communication only

Definition at line 121 of file default_cfe_sb_extern_typedefs.h.

11.237.2 Field Documentation

11.237.2.1 Priority `uint8 CFE_SB_Qos_t::Priority`

Specify high(1) or low(0) message priority for off-board routing, currently unused.

Definition at line 123 of file default_cfe_sb_extern_typedefs.h.

11.237.2.2 Reliability `uint8 CFE_SB_Qos_t::Reliability`

Specify high(1) or low(0) message transfer reliability for off-board routing, currently unused.

Definition at line 124 of file default_cfe_sb_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h

11.238 CFE_SB_ResetCountersCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

11.238.1 Detailed Description

Definition at line 57 of file default_cfe_sb_msgstruct.h.

11.238.2 Field Documentation**11.238.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd::CommandHeader`

Definition at line 59 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.239 CFE_SB_RouteCmd_Payload Struct Reference

Enable/Disable Route Command Payload.

```
#include <default_cfe_sb_msgdefs.h>
```

Data Fields

- `CFE_SB_MsgId_t MsgId`
Message ID of route to be enabled or disabled `CFE_SB_MsgId_t`.
- `CFE_SB_Pipeld_t Pipe`
Pipe ID of route to be enabled or disabled `CFE_SB_Pipeld_t`.
- `uint8 Spare`
Spare byte to make command even number of bytes.

11.239.1 Detailed Description

Enable/Disable Route Command Payload.

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 Pipeld combination.

Definition at line 53 of file default_cfe_sb_msgdefs.h.

11.239.2 Field Documentation

11.239.2.1 MsgId [CFE_SB_MsgId_t](#) CFE_SB_RouteCmd_Payload::MsgId
Message ID of route to be enabled or disabled [CFE_SB_MsgId_t](#).
Definition at line 55 of file default_cfe_sb_msgdefs.h.

11.239.2.2 Pipe [CFE_SB_PipeId_t](#) CFE_SB_RouteCmd_Payload::Pipe
Pipe ID of route to be enabled or disabled [CFE_SB_PipeId_t](#).
Definition at line 56 of file default_cfe_sb_msgdefs.h.

11.239.2.3 Spare [uint8](#) CFE_SB_RouteCmd_Payload::Spare
Spare byte to make command even number of bytes.
Definition at line 57 of file default_cfe_sb_msgdefs.h.
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.240 CFE_SB_RoutingFileEntry Struct Reference

SB Routing File Entry.

```
#include <default_cfe_sb_msgdefs.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.

11.240.1 Detailed Description

SB Routing File Entry.

Structure of one element of the routing information in response to [CFE_SB_WRITE_ROUTING_INFO_CC](#)
Definition at line 211 of file default_cfe_sb_msgdefs.h.

11.240.2 Field Documentation

11.240.2.1 AppName char CFE_SB_RoutingFileEntry::AppName [[CFE_MISSION_MAX_API_LEN](#)]
Pipe Depth Statistics.
Definition at line 217 of file default_cfe_sb_msgdefs.h.

11.240.2.2 MsgCnt `uint16 CFE_SB_RoutingFileEntry::MsgCnt`

Number of msgs with this MsgId sent to this PipeId.

Definition at line 216 of file default_cfe_sb_msgdefs.h.

11.240.2.3 MsgId `CFE_SB_MsgId_t CFE_SB_RoutingFileEntry::MsgId`

Message Id portion of the route.

Definition at line 213 of file default_cfe_sb_msgdefs.h.

11.240.2.4 PipeId `CFE_SB_PipeId_t CFE_SB_RoutingFileEntry::PipeId`

Pipe Id portion of the route.

Definition at line 214 of file default_cfe_sb_msgdefs.h.

11.240.2.5 PipeName `char CFE_SB_RoutingFileEntry::PipeName[CFE_MISSION_MAX_API_LEN]`

Pipe Depth Statistics.

Definition at line 218 of file default_cfe_sb_msgdefs.h.

11.240.2.6 State `uint8 CFE_SB_RoutingFileEntry::State`

Route Enabled or Disabled.

Definition at line 215 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.241 CFE_SB_SendHkCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

11.241.1 Detailed Description

Definition at line 82 of file default_cfe_sb_msgstruct.h.

11.241.2 Field Documentation

11.241.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_SB_SendHkCmd::CommandHeader`

Definition at line 84 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.242 CFE_SB_SendPrevSubsCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

11.242.1 Detailed Description

Definition at line 77 of file default_cfe_sb_msgstruct.h.

11.242.2 Field Documentation**11.242.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_SB_SendPrevSubsCmd::CommandHeader**

Definition at line 79 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/[default_cfe_sb_msgstruct.h](#)

11.243 CFE_SB_SendSbStatsCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

11.243.1 Detailed Description

Definition at line 72 of file default_cfe_sb_msgstruct.h.

11.243.2 Field Documentation**11.243.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_SB_SendSbStatsCmd::CommandHeader**

Definition at line 74 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/[default_cfe_sb_msgstruct.h](#)

11.244 CFE_SB_SingleSubscriptionTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_SB_SingleSubscriptionTlm_Payload_t](#) Payload
Telemetry payload.

11.244.1 Detailed Description

Definition at line 139 of file default_cfe_sb_msgstruct.h.

11.244.2 Field Documentation

11.244.2.1 Payload [CFE_SB_SingleSubscriptionTlm_Payload_t](#) CFE_SB_SingleSubscriptionTlm::Payload
Telemetry payload.

Definition at line 142 of file default_cfe_sb_msgstruct.h.

11.244.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CFE_SB_SingleSubscriptionTlm::Telemetry↔
Header

Telemetry header.

Definition at line 141 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.245 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.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

11.245.1 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 242 of file default_cfe_sb_msgdefs.h.

11.245.2 Field Documentation

11.245.2.1 MsgId [CFE_SB_MsgId_t](#) CFE_SB_SingleSubscriptionTlm_Payload::MsgId
MsgId subscribed or unsubscribe to.

Definition at line 245 of file default_cfe_sb_msgdefs.h.

11.245.2.2 Pipe `CFE_SB_PipeId_t` `CFE_SB_SingleSubscriptionTlm_Payload::Pipe`
Destination pipe id to send above msg id

Definition at line 247 of file default_cfe_sb_msgdefs.h.

11.245.2.3 Qos `CFE_SB_Qos_t` `CFE_SB_SingleSubscriptionTlm_Payload::Qos`
Quality of Service, used only for interprocessor communication.

Definition at line 246 of file default_cfe_sb_msgdefs.h.

11.245.2.4 SubType `uint8` `CFE_SB_SingleSubscriptionTlm_Payload::SubType`
Subscription or Unsubscription.

Definition at line 244 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgdefs.h`

11.246 CFE_SB_StatsTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_SB_StatsTlm_Payload_t Payload`
Telemetry payload.

11.246.1 Detailed Description

Definition at line 133 of file default_cfe_sb_msgstruct.h.

11.246.2 Field Documentation

11.246.2.1 Payload `CFE_SB_StatsTlm_Payload_t` `CFE_SB_StatsTlm::Payload`
Telemetry payload.

Definition at line 136 of file default_cfe_sb_msgstruct.h.

11.246.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_SB_StatsTlm::TelemetryHeader`
Telemetry header.

Definition at line 135 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

11.247 CFE_SB_StatsTlm_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.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**
Maximum allowed pipe depth.
- **CFE_SB_PipeDepthStats_t PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]**
Pipe Depth Statistics [CFE_SB_PipeDepthStats_t](#).

11.247.1 Detailed Description

Name SB Statistics Telemetry Packet

SB Statistics packet sent in response to [CFE_SB_SEND_SB_STATS_CC](#)
Definition at line 163 of file default_cfe_sb_msgdefs.h.

11.247.2 Field Documentation

11.247.2.1 MaxMemAllowed `uint32 CFE_SB_StatsTlm_Payload::MaxMemAllowed`
cFE Cfg Param [CFE_PLATFORM_SB_BUF_MEMORY_BYTES](#)

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMMBMALW

Definition at line 183 of file default_cfe_sb_msgdefs.h.

11.247.2.2 MaxMsgIdsAllowed `uint32` `CFE_SB_StatsTlm_Payload::MaxMsgIdsAllowed`
cFE Cfg Param `CFE_PLATFORM_SB_MAX_MSG_IDS`

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMMMDALW`

Definition at line 169 of file `default_cfe_sb_msgdefs.h`.

11.247.2.3 MaxPipeDepthAllowed `uint32` `CFE_SB_StatsTlm_Payload::MaxPipeDepthAllowed`
Maximum allowed pipe depth.

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMMPDALW`

Definition at line 199 of file `default_cfe_sb_msgdefs.h`.

11.247.2.4 MaxPipesAllowed `uint32` `CFE_SB_StatsTlm_Payload::MaxPipesAllowed`
cFE Cfg Param `CFE_PLATFORM_SB_MAX_PIPES`

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMMPALW`

Definition at line 176 of file `default_cfe_sb_msgdefs.h`.

11.247.2.5 MaxSubscriptionsAllowed `uint32` `CFE_SB_StatsTlm_Payload::MaxSubscriptionsAllowed`
product of `CFE_PLATFORM_SB_MAX_MSG_IDS` and `CFE_PLATFORM_SB_MAX_DEST_PER_PKT`

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMMSALW`

Definition at line 190 of file `default_cfe_sb_msgdefs.h`.

11.247.2.6 MemInUse `uint32` `CFE_SB_StatsTlm_Payload::MemInUse`
Memory bytes currently in use for SB msg transfers.

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMBMIU`

Definition at line 179 of file `default_cfe_sb_msgdefs.h`.

11.247.2.7 MsgIdsInUse `uint32` `CFE_SB_StatsTlm_Payload::MsgIdsInUse`
Current number of MsgIds with a destination.

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMMIDIU`

Definition at line 165 of file `default_cfe_sb_msgdefs.h`.

11.247.2.8 PeakMemInUse `uint32` `CFE_SB_StatsTlm_Payload::PeakMemInUse`
Peak memory bytes in use for SB msg transfers.

Telemetry Mnemonic(s) `$sc_$cpu_SB_Stat.SB_SMPBMIU`

Definition at line 181 of file `default_cfe_sb_msgdefs.h`.

11.247.2.9 PeakMsgIdsInUse `uint32 CFE_SB_StatsTlm_Payload::PeakMsgIdsInUse`
Peak number of MsgIds with a destination.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPMIDIU

Definition at line 167 of file default_cfe_sb_msgdefs.h.

11.247.2.10 PeakPipesInUse `uint32 CFE_SB_StatsTlm_Payload::PeakPipesInUse`
Peak number of pipes since last reboot.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPPIU

Definition at line 174 of file default_cfe_sb_msgdefs.h.

11.247.2.11 PeakSBBuffersInUse `uint32 CFE_SB_StatsTlm_Payload::PeakSBBuffersInUse`
Max number of SB message buffers in use.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPSBBIU

Definition at line 196 of file default_cfe_sb_msgdefs.h.

11.247.2.12 PeakSubscriptionsInUse `uint32 CFE_SB_StatsTlm_Payload::PeakSubscriptionsInUse`
Peak number of subscriptions.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPSIU

Definition at line 188 of file default_cfe_sb_msgdefs.h.

11.247.2.13 PipeDepthStats `CFE_SB_PipeDepthStats_t CFE_SB_StatsTlm_Payload::PipeDepthStats[CFE_MISSION_SB_MAX_PIPES]`
Pipe Depth Statistics `CFE_SB_PipeDepthStats_t`.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES]

Definition at line 202 of file default_cfe_sb_msgdefs.h.

11.247.2.14 PipesInUse `uint32 CFE_SB_StatsTlm_Payload::PipesInUse`
Number of pipes currently in use.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPIU

Definition at line 172 of file default_cfe_sb_msgdefs.h.

11.247.2.15 SBBuffersInUse `uint32 CFE_SB_StatsTlm_Payload::SBBuffersInUse`
Number of SB message buffers currently in use.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMSBBIU

Definition at line 194 of file default_cfe_sb_msgdefs.h.

11.247.2.16 SubscriptionsInUse `uint32 CFE_SB_StatsTlm_Payload::SubscriptionsInUse`
Number of current subscriptions.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMSIU

Definition at line 186 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.248 CFE_SB_SubEntries Struct Reference

SB Previous Subscriptions Entry.

```
#include <default_cfe_sb_msgdefs.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.

11.248.1 Detailed Description

SB Previous Subscriptions Entry.

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 258 of file default_cfe_sb_msgdefs.h.

11.248.2 Field Documentation

11.248.2.1 MsgId `CFE_SB_MsgId_t CFE_SB_SubEntries::MsgId`

MsgId portion of the subscription.

Definition at line 260 of file default_cfe_sb_msgdefs.h.

11.248.2.2 Pipe `CFE_SB_PipeId_t CFE_SB_SubEntries::Pipe`

PipeId portion of the subscription.

Definition at line 262 of file default_cfe_sb_msgdefs.h.

11.248.2.3 Qos `CFE_SB_Qos_t CFE_SB_SubEntries::Qos`

Qos portion of the subscription.

Definition at line 261 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.249 CFE_SB_WriteFileInfoCmd_Payload Struct Reference

Write File Info Command Payload.

```
#include <default_cfe_sb_msgdefs.h>
```

Data Fields

- char [Filename \[CFE_MISSION_MAX_PATH_LEN\]](#)

Path and Filename of data to be loaded.

11.249.1 Detailed Description

Write File Info Command Payload.

This structure contains a generic definition used by SB commands that write to a file

Definition at line 40 of file default_cfe_sb_msgdefs.h.

11.249.2 Field Documentation

11.249.2.1 **Filename** `char CFE_SB_WriteFileInfoCmd_Payload::Filename [CFE_MISSION_MAX_PATH_LEN]`

Path and Filename of data to be loaded.

Definition at line 42 of file default_cfe_sb_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgdefs.h

11.250 CFE_SB_WriteMapInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)

Command header.

- [CFE_SB_WriteFileInfoCmd_Payload_t Payload](#)

Command payload.

11.250.1 Detailed Description

Definition at line 102 of file default_cfe_sb_msgstruct.h.

11.250.2 Field Documentation

11.250.2.1 **CommandHeader** `CFE_MSG_CommandHeader_t CFE_SB_WriteMapInfoCmd::CommandHeader`

Command header.

Definition at line 104 of file default_cfe_sb_msgstruct.h.

11.250.2.2 Payload `CFE_SB_WriteFileInfoCmd_Payload_t` `CFE_SB_WriteMapInfoCmd::Payload`
Command payload.

Definition at line 105 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgstruct.h](#)

11.251 CFE_SB_WritePipeInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_SB_WriteFileInfoCmd_Payload_t Payload`
Command payload.

11.251.1 Detailed Description

Definition at line 96 of file default_cfe_sb_msgstruct.h.

11.251.2 Field Documentation

11.251.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_SB_WritePipeInfoCmd::CommandHeader`
Command header.

Definition at line 98 of file default_cfe_sb_msgstruct.h.

11.251.2.2 Payload `CFE_SB_WriteFileInfoCmd_Payload_t` `CFE_SB_WritePipeInfoCmd::Payload`
Command payload.

Definition at line 99 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default_cfe_sb_msgstruct.h](#)

11.252 CFE_SB_WriteRoutingInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_SB_WriteFileInfoCmd_Payload_t Payload`
Command payload.

11.252.1 Detailed Description

Definition at line 90 of file default_cfe_sb_msgstruct.h.

11.252.2 Field Documentation

11.252.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_SB_WriteRoutingInfoCmd::CommandHeader
Command header.

Definition at line 92 of file default_cfe_sb_msgstruct.h.

11.252.2.2 Payload [CFE_SB_WriteFileInfoCmd_Payload_t](#) CFE_SB_WriteRoutingInfoCmd::Payload
Command payload.

Definition at line 93 of file default_cfe_sb_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default_cfe_sb_msgstruct.h

11.253 CFE_TBL_AbortLoadCmd Struct Reference

Abort Load Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_AbortLoadCmd_Payload_t](#) Payload
Command payload.

11.253.1 Detailed Description

Abort Load Command.

Definition at line 131 of file default_cfe_tbl_msgstruct.h.

11.253.2 Field Documentation

11.253.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_AbortLoadCmd::CommandHeader
Command header.

Definition at line 133 of file default_cfe_tbl_msgstruct.h.

11.253.2.2 Payload [CFE_TBL_AbortLoadCmd_Payload_t](#) CFE_TBL_AbortLoadCmd::Payload
Command payload.

Definition at line 134 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.254 CFE_TBL_AbortLoadCmd_Payload Struct Reference

Abort Load Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)
Full Name of Table whose load is to be aborted.

11.254.1 Detailed Description

Abort Load Command Payload.

For command details, see [CFE_TBL_ABORT_LOAD_CC](#)

Definition at line 148 of file default_cfe_tbl_msgdefs.h.

11.254.2 Field Documentation

11.254.2.1 TableName char CFE_TBL_AbortLoadCmd_Payload::TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

Full Name of Table whose load is to be aborted.

ASCII string containing full table name identifier of a table whose load is to be aborted

Definition at line 150 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.255 CFE_TBL_ActivateCmd Struct Reference

Activate Table Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TBL_ActivateCmd_Payload_t Payload](#)
Command payload.

11.255.1 Detailed Description

Activate Table Command.

Definition at line 95 of file default_cfe_tbl_msgstruct.h.

11.255.2 Field Documentation

11.255.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_ActivateCmd::CommandHeader

Command header.

Definition at line 97 of file default_cfe_tbl_msgstruct.h.

11.255.2.2 Payload [CFE_TBL_ActivateCmd_Payload_t](#) CFE_TBL_ActivateCmd::Payload

Command payload.

Definition at line 98 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.256 CFE_TBL_ActivateCmd_Payload Struct Reference

Activate Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Full Name of Table to be activated.

11.256.1 Detailed Description

Activate Table Command Payload.

For command details, see [CFE_TBL_ACTIVATE_CC](#)

Definition at line 95 of file default_cfe_tbl_msgdefs.h.

11.256.2 Field Documentation

11.256.2.1 TableName char CFE_TBL_ActivateCmd_Payload::TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

Full Name of Table to be activated.

ASCII string containing full table name identifier of table to be activated

Definition at line 97 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.257 CFE_TBL_DelCDSCmd_Payload Struct Reference

Delete Critical Table CDS Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Full Name of Table whose CDS is to be deleted.

11.257.1 Detailed Description

Delete Critical Table CDS Command Payload.

For command details, see [CFE_TBL_DELETE_CDS_CC](#)

Definition at line 134 of file default_cfe_tbl_msgdefs.h.

11.257.2 Field Documentation

11.257.2.1 TableName char CFE_TBL_DelCDSCmd_Payload::TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

Full Name of Table whose CDS is to be deleted.

ASCII string containing full table name identifier of a critical table whose CDS is to be deleted

Definition at line 136 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.258 CFE_TBL_DeleteCDSCmd Struct Reference

Delete Critical Table CDS Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_DelCDSCmd_Payload_t](#) Payload
Command payload.

11.258.1 Detailed Description

Delete Critical Table CDS Command.

Definition at line 122 of file default_cfe_tbl_msgstruct.h.

11.258.2 Field Documentation

11.258.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_DeleteCDSCmd::CommandHeader

Command header.

Definition at line 124 of file default_cfe_tbl_msgstruct.h.

11.258.2.2 Payload [CFE_TBL_DelCDSCmd_Payload_t](#) CFE_TBL_DeleteCDSCmd::Payload

Command payload.

Definition at line 125 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.259 CFE_TBL_DumpCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_DumpCmd_Payload_t](#) Payload
Command payload.

11.259.1 Detailed Description

/brief Dump Table Command

Definition at line 77 of file default_cfe_tbl_msgstruct.h.

11.259.2 Field Documentation

11.259.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TBL_DumpCmd::CommandHeader`
Command header.
Definition at line 79 of file default_cfe_tbl_msgstruct.h.

11.259.2.2 Payload `CFE_TBL_DumpCmd_Payload_t` `CFE_TBL_DumpCmd::Payload`
Command payload.
Definition at line 80 of file default_cfe_tbl_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.260 CFE_TBL_DumpCmd_Payload Struct Reference

Dump Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- `CFE_TBL_BufferSelect_Enum_t 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.

11.260.1 Detailed Description

Dump Table Command Payload.

For command details, see [CFE_TBL_DUMP_CC](#)

Definition at line 56 of file default_cfe_tbl_msgdefs.h.

11.260.2 Field Documentation

11.260.2.1 ActiveTableFlag `CFE_TBL_BufferSelect_Enum_t` `CFE_TBL_DumpCmd_Payload::ActiveTableFlag`
`CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table`
Selects either the "Inactive" (`CFE_TBL_BufferSelect_INACTIVE`) buffer or the "Active" (`CFE_TBL_BufferSelect_ACTIVE`)
buffer to be dumped
Definition at line 58 of file default_cfe_tbl_msgdefs.h.

11.260.2.2 DumpFilename `char` `CFE_TBL_DumpCmd_Payload::DumpFilename [CFE_MISSION_MAX_PATH_LEN]`
Full Filename where data is to be written.
ASCII string containing full path of filename where data is to be dumped
Definition at line 67 of file default_cfe_tbl_msgdefs.h.

11.260.2.3 TableName char CFE_TBL_DumpCmd_Payload::TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
Full name of table to be dumped.

ASCII string containing full table name identifier of table to be dumped
Definition at line 64 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.261 CFE_TBL_DumpRegistryCmd Struct Reference

Dump Registry Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t** CommandHeader
Command header.
- **CFE_TBL_DumpRegistryCmd_Payload_t** Payload
Command payload.

11.261.1 Detailed Description

Dump Registry Command.

Definition at line 104 of file default_cfe_tbl_msgstruct.h.

11.261.2 Field Documentation

11.261.2.1 CommandHeader **CFE_MSG_CommandHeader_t** CFE_TBL_DumpRegistryCmd::CommandHeader
Command header.

Definition at line 106 of file default_cfe_tbl_msgstruct.h.

11.261.2.2 Payload **CFE_TBL_DumpRegistryCmd_Payload_t** CFE_TBL_DumpRegistryCmd::Payload
Command payload.

Definition at line 107 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.262 CFE_TBL_DumpRegistryCmd_Payload Struct Reference

Dump Registry Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char DumpFilename [CFE_MISSION_MAX_PATH_LEN]
Full Filename where dumped data is to be written.

11.262.1 Detailed Description

Dump Registry Command Payload.

For command details, see [CFE_TBL_DUMP_REGISTRY_CC](#)

Definition at line 107 of file default_cfe_tbl_msgdefs.h.

11.262.2 Field Documentation

11.262.2.1 DumpFilename `char CFE_TBL_DumpRegistryCmd_Payload::DumpFilename[CFE_MISSION_MAX_PATH_LEN]`

Full Filename where dumped data is to be written.

ASCII string containing full path of filename where registry is to be dumped

Definition at line 109 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

11.263 CFE_TBL_File_Hdr Struct Reference

The definition of the header fields that are included in CFE Table Data files.

```
#include <default_cfe_tbl_extern_typedefs.h>
```

Data Fields

- `uint32 Reserved`
- `uint32 Offset`
- `uint32 NumBytes`
- `char TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

11.263.1 Detailed Description

The definition of the header fields that are included in CFE Table Data files.

This header follows the CFE_FS header and precedes the actual table data.

Note

The Offset and NumBytes fields in the table header are 32 bits for backward compatibility with existing CFE versions. This means that even on 64-bit CPUs, individual table files will be limited to 4GiB in size.

Definition at line 64 of file `default_cfe_tbl_extern_typedefs.h`.

11.263.2 Field Documentation

11.263.2.1 NumBytes `uint32 CFE_TBL_File_Hdr::NumBytes`

Number of bytes to load into table

Definition at line 68 of file `default_cfe_tbl_extern_typedefs.h`.

11.263.2.2 Offset `uint32 CFE_TBL_File_Hdr::Offset`

Byte Offset at which load should commence

Definition at line 67 of file `default_cfe_tbl_extern_typedefs.h`.

11.263.2.3 Reserved `uint32 CFE_TBL_File_Hdr::Reserved`

Future Use: NumTblSegments in File?

Definition at line 66 of file `default_cfe_tbl_extern_typedefs.h`.

11.263.2.4 TableName char CFE_TBL_File_Hdr::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

Fully qualified name of table to load

Definition at line 69 of file default_cfe_tbl_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h

11.264 CFE_TBL_FileDef Struct Reference

Table File summary object.

```
#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.

11.264.1 Detailed Description

Table File summary object.

The definition of the file definition metadata that can be used by external tools (e.g. elf2cfetbl) to generate CFE table data files.

Definition at line 58 of file cfe_tbl_filedef.h.

11.264.2 Field Documentation

11.264.2.1 Description

 char CFE_TBL_FileDef::Description[CFE_FS_HDR_DESC_MAX_LEN]

Description of table image that is included in cFE File Header.

This is a free-form text string that can be any meaningful value

Definition at line 94 of file cfe_tbl_filedef.h.

11.264.2.2 ObjectName

 char CFE_TBL_FileDef::ObjectName[64]

Name of instantiated variable that contains desired table image.

Note

For consistency and future compatibility with auto-generated table files and table definitions, the "ObjectName" should match the table struct typedef name without the "_t" suffix. For example, the limit checker action table (ADT) is defined by a type called "LC_ADT_t", the ObjectName should be "LC_ADT".

This naming convention allows the type name to be inferred from the ObjectName (and vice-versa) without having to directly specify both the type name and object name here.

Although the traditional elf2cfcfetbl tool does not currently do any type checking, future tool versions may add more robust type verification and therefore need to know the type name as well as the object name.

Definition at line 76 of file cfe_tbl_filedef.h.

11.264.2.3 ObjectSize `uint32 CFE_TBL_FileDef::ObjectSize`

Size, in bytes, of instantiated object.

This may be used by tools to check for consistency between the actual defined table size and the expected table size.

This is set automatically via the `CFE_TBL_FILEDEF` macro.

Definition at line 112 of file cfe_tbl_filedef.h.

11.264.2.4 TableName `char CFE_TBL_FileDef::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

Name of Table as defined onboard.

This should be in the form of "APP_NAME.TABLE_NAME" where APP_NAME matches what the app is named at runtime (the 4th column of cfe_es_startup.scr) and TABLE_NAME matches the 2nd parameter of the call to `CFE_TBL_Register()`. Preferably the TABLE_NAME should also match the ObjectName here in this structure, although this is not strictly required, it helps keep things consistent.

Definition at line 87 of file cfe_tbl_filedef.h.

11.264.2.5 TgtFilename `char CFE_TBL_FileDef::TgtFilename[CFE_MISSION_MAX_FILE_LEN]`

Default filename to be used for output of elf2cfcfetbl utility.

This must match the expected table file name, which is the name of the source file but the ".c" extension replaced with ".tbl". This is the filename only - do not include a directory/path name here, it can be copied to any runtime directory on the target by external scripts, but should not be renamed.

Definition at line 104 of file cfe_tbl_filedef.h.

The documentation for this struct was generated from the following file:

- cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h

11.265 CFE_TBL_HousekeepingTlm Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_TBL_HousekeepingTlm_Payload_t Payload`
Telemetry payload.

11.265.1 Detailed Description

Definition at line 154 of file default_cfe_tbl_msgstruct.h.

11.265.2 Field Documentation

11.265.2.1 Payload `CFE_TBL_HousekeepingTlm_Payload_t CFE_TBL_HousekeepingTlm::Payload`

Telemetry payload.

Definition at line 157 of file default_cfe_tbl_msgstruct.h.

11.265.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_TBL_HousekeepingTlm::TelemetryHeader`
Telemetry header.

Definition at line 156 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

11.266 CFE_TBL_HousekeepingTlm_Payload Struct Reference

```
#include <default_cfe_tbl_msgdefs.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.

11.266.1 Detailed Description

Name Table Services Housekeeping Packet

Definition at line 179 of file default_cfe_tbl_msgdefs.h.

11.266.2 Field Documentation

11.266.2.1 ActiveBuffer bool CFE_TBL_HousekeepingTlm_Payload::ActiveBuffer
Indicator of whether table buffer validated was 0=Inactive, 1=Active.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastValBuf

Definition at line 206 of file default_cfe_tbl_msgdefs.h.

11.266.2.2 ByteAlignPad1 uint8 CFE_TBL_HousekeepingTlm_Payload::ByteAlignPad1
Spare byte to ensure longword alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_BytAlignPad1

Definition at line 222 of file default_cfe_tbl_msgdefs.h.

11.266.2.3 CommandCounter uint8 CFE_TBL_HousekeepingTlm_Payload::CommandCounter
Count of valid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CMDPC

Definition at line 184 of file default_cfe_tbl_msgdefs.h.

11.266.2.4 CommandErrorCounter uint8 CFE_TBL_HousekeepingTlm_Payload::CommandErrorCounter
Count of invalid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CMDEC

Definition at line 186 of file default_cfe_tbl_msgdefs.h.

11.266.2.5 FailedValCounter uint8 CFE_TBL_HousekeepingTlm_Payload::FailedValCounter
Total number of unsuccessful table validations.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ValFailedCtr

Definition at line 212 of file default_cfe_tbl_msgdefs.h.

11.266.2.6 LastFileDumped char CFE_TBL_HousekeepingTlm_Payload::LastFileDumped[[CFE_MISSION_MAX_PATH_LEN](#)]
Path and Name of last file dumped to.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastFileDumped[OS_MAX_PATH_LEN]

Definition at line 232 of file default_cfe_tbl_msgdefs.h.

11.266.2.7 LastFileLoaded char CFE_TBL_HousekeepingTlm_Payload::LastFileLoaded[[CFE_MISSION_MAX_PATH_LEN](#)]
Path and Name of last table image file loaded.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastFileLoaded[OS_MAX_PATH_LEN]

Definition at line 230 of file default_cfe_tbl_msgdefs.h.

11.266.2.8 LastTableLoaded char CFE_TBL_HousekeepingTlm_Payload::LastTableLoaded[[CFE_MISSION_TBL_MAX_FULL_NAME_LEN](#)]
Name of the last table loaded.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastTableLoaded[CFE_TBL_MAX_FULL_NAME_LEN]

Definition at line 234 of file default_cfe_tbl_msgdefs.h.

11.266.2.9 LastUpdatedTable char CFE_TBL_HousekeepingTlm_Payload::LastUpdatedTable[[CFE_MISSION_TBL_MAX_FULL_NAME_LEN](#)]
Name of the last table updated.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastUpdTblName[CFE_TB_MAX_FULL_NAME_LEN]

Definition at line 228 of file default_cfe_tbl_msgdefs.h.

11.266.2.10 LastUpdateTime [CFE_TIME_SysTime_t](#) CFE_TBL_HousekeepingTlm_Payload::LastUpdateTime
Time of last table update.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastUpdTIme, \$sc_\$cpu_TBL_SECONDS, \$sc_\$cpu_TBL_SUBSECONDS

Definition at line 226 of file default_cfe_tbl_msgdefs.h.

11.266.2.11 LastValCrc [uint32](#) CFE_TBL_HousekeepingTlm_Payload::LastValCrc
Data Integrity Value computed for last table validated.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastValCRC

Definition at line 202 of file default_cfe_tbl_msgdefs.h.

11.266.2.12 LastValStatus [int32](#) CFE_TBL_HousekeepingTlm_Payload::LastValStatus
Returned status from validation function for last table validated.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastValS

Definition at line 204 of file default_cfe_tbl_msgdefs.h.

11.266.2.13 LastValTableName `char CFE_TBL_HousekeepingTlm_Payload::LastValTableName[CFE_MISSION_TBL_MAX_FULL_NAME]`
Name of last table validated.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]`

Definition at line 208 of file default_cfe_tbl_msgdefs.h.

11.266.2.14 MemPoolHandle `CFE_ES_MemHandle_t CFE_TBL_HousekeepingTlm_Payload::MemPoolHandle`
Handle to TBL's memory pool.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_MemPoolHandle`

Definition at line 224 of file default_cfe_tbl_msgdefs.h.

11.266.2.15 NumFreeSharedBufs `uint8 CFE_TBL_HousekeepingTlm_Payload::NumFreeSharedBufs`
Number of free Shared Working Buffers.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_NumFreeShrBuf`

Definition at line 220 of file default_cfe_tbl_msgdefs.h.

11.266.2.16 NumLoadPending `uint16 CFE_TBL_HousekeepingTlm_Payload::NumLoadPending`
Number of Tables pending on Applications for their update.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_NumUpdatesPend`

Definition at line 194 of file default_cfe_tbl_msgdefs.h.

11.266.2.17 NumTables `uint16 CFE_TBL_HousekeepingTlm_Payload::NumTables`
Number of Tables Registered.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_NumTables`

Definition at line 192 of file default_cfe_tbl_msgdefs.h.

11.266.2.18 NumValRequests `uint8 CFE_TBL_HousekeepingTlm_Payload::NumValRequests`
Number of times Table Services has requested validations from Apps.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_ValReqCtr`

Definition at line 214 of file default_cfe_tbl_msgdefs.h.

11.266.2.19 SuccessValCounter `uint8 CFE_TBL_HousekeepingTlm_Payload::SuccessValCounter`
Total number of successful table validations.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_ValSuccessCtr`

Definition at line 210 of file default_cfe_tbl_msgdefs.h.

11.266.2.20 ValidationCounter `uint16` `CFE_TBL_HousekeepingTlm_Payload::ValidationCounter`
Number of completed table validations.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ValCompltdCtr

Definition at line 200 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.267 CFE_TBL_Info Struct Reference

Table Info.

```
#include <cfef_tbl_api_typedefs.h>
```

Data Fields

- `size_t Size`
Size, in bytes, of Table.
- `uint32 NumUsers`
Number of Apps with access to the table.
- `CFE_TIME_SysTime_t FileTime`
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 [CFE_MISSION_MAX_PATH_LEN]`
Filename of last file loaded into table.

11.267.1 Detailed Description

Table Info.

Definition at line 109 of file cfe_tbl_api_typedefs.h.

11.267.2 Field Documentation

11.267.2.1 Crc `uint32` `CFE_TBL_Info::Crc`

Most recently calculated CRC by TBL services on table contents.

Definition at line 114 of file cfe_tbl_api_typedefs.h.

11.267.2.2 Critical `bool CFE_TBL_Info::Critical`
Flag indicating Table contents are maintained in a CDS.
Definition at line 120 of file `cfe_tbl_api_typedefs.h`.

11.267.2.3 DoubleBuffered `bool CFE_TBL_Info::DoubleBuffered`
Flag indicating Table has a dedicated inactive buffer.
Definition at line 118 of file `cfe_tbl_api_typedefs.h`.

11.267.2.4 DumpOnly `bool CFE_TBL_Info::DumpOnly`
Flag indicating Table is NOT to be loaded.
Definition at line 117 of file `cfe_tbl_api_typedefs.h`.

11.267.2.5 FileTime `CFE_TIME_SysTime_t CFE_TBL_Info::FileTime`
File creation time from last file loaded into table.
Definition at line 113 of file `cfe_tbl_api_typedefs.h`.

11.267.2.6 LastFileLoaded `char CFE_TBL_Info::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]`
Filename of last file loaded into table.
Definition at line 121 of file `cfe_tbl_api_typedefs.h`.

11.267.2.7 NumUsers `uint32 CFE_TBL_Info::NumUsers`
Number of Apps with access to the table.
Definition at line 112 of file `cfe_tbl_api_typedefs.h`.

11.267.2.8 Size `size_t CFE_TBL_Info::Size`
Size, in bytes, of Table.
Definition at line 111 of file `cfe_tbl_api_typedefs.h`.

11.267.2.9 TableLoadedOnce `bool CFE_TBL_Info::TableLoadedOnce`
Flag indicating whether table has been loaded once or not.
Definition at line 116 of file `cfe_tbl_api_typedefs.h`.

11.267.2.10 TimeOfLastUpdate `CFE_TIME_SysTime_t CFE_TBL_Info::TimeOfLastUpdate`
Time when Table was last updated.
Definition at line 115 of file `cfe_tbl_api_typedefs.h`.

11.267.2.11 UserDefAddr `bool CFE_TBL_Info::UserDefAddr`
Flag indicating Table address was defined by Owner Application.
Definition at line 119 of file `cfe_tbl_api_typedefs.h`.
The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h`

11.268 CFE_TBL_LoadCmd Struct Reference

Load Table Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_LoadCmd_Payload_t](#) Payload
Command payload.

11.268.1 Detailed Description

Load Table Command.

Definition at line 68 of file default_cfe_tbl_msgstruct.h.

11.268.2 Field Documentation

11.268.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_LoadCmd::CommandHeader

Command header.

Definition at line 70 of file default_cfe_tbl_msgstruct.h.

11.268.2.2 Payload [CFE_TBL_LoadCmd_Payload_t](#) CFE_TBL_LoadCmd::Payload

Command payload.

Definition at line 71 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.269 CFE_TBL_LoadCmd_Payload Struct Reference

Load Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char LoadFilename [[CFE_MISSION_MAX_PATH_LEN](#)]
Filename (and path) of data to be loaded.

11.269.1 Detailed Description

Load Table Command Payload.

For command details, see [CFE_TBL_LOAD_CC](#)

Definition at line 46 of file default_cfe_tbl_msgdefs.h.

11.269.2 Field Documentation

11.269.2.1 LoadFilename `char CFE_TBL_LoadCmd_Payload::LoadFilename[CFE_MISSION_MAX_PATH_LEN]`
Filename (and path) of data to be loaded.

Definition at line 48 of file default_cfe_tbl_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h](#)

11.270 CFE_TBL_NoopCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.

11.270.1 Detailed Description

Definition at line 50 of file default_cfe_tbl_msgstruct.h.

11.270.2 Field Documentation

11.270.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_NoopCmd::CommandHeader
Command header.

Definition at line 54 of file default_cfe_tbl_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h](#)

11.271 CFE_TBL_NotifyCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TBL_NotifyCmd_Payload_t Payload](#)
Command payload.

11.271.1 Detailed Description

/brief Table Management Notification Command

Definition at line 142 of file default_cfe_tbl_msgstruct.h.

11.271.2 Field Documentation

11.271.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_NotifyCmd::CommandHeader
Command header.

Definition at line 144 of file default_cfe_tbl_msgstruct.h.

11.271.2.2 Payload `CFE_TBL_NotifyCmd_Payload_t` `CFE_TBL_NotifyCmd::Payload`
Command payload.

Definition at line 145 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

11.272 CFE_TBL_NotifyCmd_Payload Struct Reference

Table Management Notification Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- `uint32 Parameter`

Application specified command parameter.

11.272.1 Detailed Description

Table Management Notification Command Payload.

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 166 of file `default_cfe_tbl_msgdefs.h`.

11.272.2 Field Documentation

11.272.2.1 Parameter `uint32 CFE_TBL_NotifyCmd_Payload::Parameter`

Application specified command parameter.

Definition at line 168 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

11.273 CFE_TBL_ResetCountersCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

Command header.

11.273.1 Detailed Description

Definition at line 55 of file `default_cfe_tbl_msgstruct.h`.

11.273.2 Field Documentation

11.273.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TBL_ResetCountersCmd::CommandHeader`
Command header.

Definition at line 57 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

11.274 CFE_TBL_SendHkCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.274.1 Detailed Description

Definition at line 60 of file `default_cfe_tbl_msgstruct.h`.

11.274.2 Field Documentation

11.274.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TBL_SendHkCmd::CommandHeader`
Command header.

Definition at line 62 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

11.275 CFE_TBL_SendRegistryCmd Struct Reference

Send Table Registry Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TBL_SendRegistryCmd_Payload_t Payload`
Command payload.

11.275.1 Detailed Description

Send Table Registry Command.

Definition at line 113 of file `default_cfe_tbl_msgstruct.h`.

11.275.2 Field Documentation

11.275.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_SendRegistryCmd::CommandHeader
Command header.
Definition at line 115 of file default_cfe_tbl_msgstruct.h.

11.275.2.2 Payload [CFE_TBL_SendRegistryCmd_Payload_t](#) CFE_TBL_SendRegistryCmd::Payload
Command payload.
Definition at line 116 of file default_cfe_tbl_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.276 CFE_TBL_SendRegistryCmd_Payload Struct Reference

Send Table Registry Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- char **TableName** [[CFE_MISSION_TBL_MAX_FULL_NAME_LEN](#)]
Full Name of Table whose registry entry is to be telemetered.

11.276.1 Detailed Description

Send Table Registry Command Payload.

For command details, see [CFE_TBL_SEND_REGISTRY_CC](#)

Definition at line 120 of file default_cfe_tbl_msgdefs.h.

11.276.2 Field Documentation

11.276.2.1 TableName char CFE_TBL_SendRegistryCmd_Payload::TableName [[CFE_MISSION_TBL_MAX_FULL_NAME_LEN](#)]
Full Name of Table whose registry entry is to be telemetered.
ASCII string containing full table name identifier of table whose registry entry is to be telemetered via
[CFE_TBL_TableRegistryTlm_t](#)
Definition at line 122 of file default_cfe_tbl_msgdefs.h.
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.277 CFE_TBL_TableRegistryTlm Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_TBL_TblRegPacket_Payload_t](#) Payload
Telemetry payload.

11.277.1 Detailed Description

Definition at line 160 of file default_cfe_tbl_msgstruct.h.

11.277.2 Field Documentation

11.277.2.1 Payload `CFE_TBL_TblRegPacket_Payload_t` `CFE_TBL_TableRegistryTlm::Payload`
Telemetry payload.

Definition at line 163 of file `default_cfe_tbl_msgstruct.h`.

11.277.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_TBL_TableRegistryTlm::TelemetryHeader`
Telemetry header.

Definition at line 162 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

11.278 CFE_TBL_TblRegPacket_Payload Struct Reference

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- **`CFE_ES_MemOffset_t` Size**
Size, in bytes, of Table.
- **`uint32 Crc`**
Most recently calculated CRC of Table.
- **`CFE_ES_MemAddress_t` ActiveBufferAddr**
Address of Active Buffer.
- **`CFE_ES_MemAddress_t` InactiveBufferAddr**
Address of Inactive Buffer.
- **`CFE_ES_MemAddress_t` ValidationFuncPtr**
Ptr to Owner App's function that validates tbl contents.
- **`CFE_TIME_SysTime_t` TimeOfLastUpdate**
Time when Table was last updated.
- **`CFE_TIME_SysTime_t` FileTime**
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.

11.278.1 Detailed Description

Name Table Registry Info Packet

Definition at line 241 of file default_cfe_tbl_msgdefs.h.

11.278.2 Field Documentation

11.278.2.1 ActiveBufferAddr `CFE_ES_MemAddress_t` `CFE_TBL_TblRegPacket_Payload::ActiveBufferAddr`
Address of Active Buffer.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ActBufAdd

Definition at line 247 of file default_cfe_tbl_msgdefs.h.

11.278.2.2 ByteAlign4 `uint8` `CFE_TBL_TblRegPacket_Payload::ByteAlign4`
Spare byte to maintain byte alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_Spare4

Definition at line 273 of file default_cfe_tbl_msgdefs.h.

11.278.2.3 Crc `uint32` `CFE_TBL_TblRegPacket_Payload::Crc`
Most recently calculated CRC of Table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CRC

Definition at line 245 of file default_cfe_tbl_msgdefs.h.

11.278.2.4 Critical `bool` `CFE_TBL_TblRegPacket_Payload::Critical`
Indicates whether table is Critical or not.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_Spare3

Definition at line 271 of file default_cfe_tbl_msgdefs.h.

11.278.2.5 DoubleBuffered `bool` `CFE_TBL_TblRegPacket_Payload::DoubleBuffered`
Flag indicating Table has a dedicated inactive buffer.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_DblBuffered

Definition at line 263 of file default_cfe_tbl_msgdefs.h.

11.278.2.6 DumpOnly bool CFE_TBL_TblRegPacket_Payload::DumpOnly
Flag indicating Table is NOT to be loaded.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_DumpOnly

Definition at line 261 of file default_cfe_tbl_msgdefs.h.

11.278.2.7 FileTime CFE_TIME_SysTime_t CFE_TBL_TblRegPacket_Payload::FileTime
File creation time from last file loaded into table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_FILECTIME

Definition at line 255 of file default_cfe_tbl_msgdefs.h.

11.278.2.8 InactiveBufferAddr CFE_ES_MemAddress_t CFE_TBL_TblRegPacket_Payload::InactiveBufferAddr
Address of Inactive Buffer.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_IActBufAdd

Definition at line 249 of file default_cfe_tbl_msgdefs.h.

11.278.2.9 LastFileLoaded char CFE_TBL_TblRegPacket_Payload::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]
Filename of last file loaded into table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastFileUpd[OS_MAX_PATH_LEN]

Definition at line 267 of file default_cfe_tbl_msgdefs.h.

11.278.2.10 LoadPending bool CFE_TBL_TblRegPacket_Payload::LoadPending
Flag indicating an inactive buffer is ready to be copied.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_UpdatePndng

Definition at line 259 of file default_cfe_tbl_msgdefs.h.

11.278.2.11 Name char CFE_TBL_TblRegPacket_Payload::Name[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
Processor specific table name.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_Name[CFE_TB_MAX_FULL_NAME_LEN]

Definition at line 265 of file default_cfe_tbl_msgdefs.h.

11.278.2.12 OwnerAppName char CFE_TBL_TblRegPacket_Payload::OwnerAppName[CFE_MISSION_MAX_API_LEN]
Name of owning application.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_OwnerApp[OS_MAX_API_NAME]

Definition at line 269 of file default_cfe_tbl_msgdefs.h.

11.278.2.13 Size `CFE_ES_MemOffset_t` `CFE_TBL_TblRegPacket_Payload::Size`
Size, in bytes, of Table.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_SIZE`

Definition at line 243 of file `default_cfe_tbl_msgdefs.h`.

11.278.2.14 TableLoadedOnce `bool` `CFE_TBL_TblRegPacket_Payload::TableLoadedOnce`
Flag indicating whether table has been loaded once or not.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_LoadedOnce`

Definition at line 257 of file `default_cfe_tbl_msgdefs.h`.

11.278.2.15 TimeOfLastUpdate `CFE_TIME_SysTime_t` `CFE_TBL_TblRegPacket_Payload::TimeOfLastUpdate`
Time when Table was last updated.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_TimeLastUpd`, `$sc_$cpu_TBL_TLUSECONDS`, `$sc_$cpu_TBL_TLUSUB←SECONDS`

Definition at line 253 of file `default_cfe_tbl_msgdefs.h`.

11.278.2.16 ValidationFuncPtr `CFE_ES_MemAddress_t` `CFE_TBL_TblRegPacket_Payload::ValidationFuncPtr`
Ptr to Owner App's function that validates tbl contents.

Telemetry Mnemonic(s) `$sc_$cpu_TBL_ValFuncPtr`

Definition at line 251 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

11.279 CFE_TBL_ValidateCmd Struct Reference

Validate Table Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TBL_ValidateCmd_Payload_t Payload`
Command payload.

11.279.1 Detailed Description

Validate Table Command.

Definition at line 86 of file `default_cfe_tbl_msgstruct.h`.

11.279.2 Field Documentation

11.279.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TBL_ValidateCmd::CommandHeader
Command header.
Definition at line 88 of file default_cfe_tbl_msgstruct.h.

11.279.2.2 Payload [CFE_TBL_ValidateCmd_Payload_t](#) CFE_TBL_ValidateCmd::Payload
Command payload.
Definition at line 89 of file default_cfe_tbl_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h

11.280 CFE_TBL_ValidateCmd_Payload Struct Reference

Validate Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

Data Fields

- [CFE_TBL_BufferSelect_Enum_t 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.

11.280.1 Detailed Description

Validate Table Command Payload.

For command details, see [CFE_TBL_VALIDATE_CC](#)

Definition at line 77 of file default_cfe_tbl_msgdefs.h.

11.280.2 Field Documentation

11.280.2.1 ActiveTableFlag [CFE_TBL_BufferSelect_Enum_t](#) CFE_TBL_ValidateCmd_Payload::ActiveTable↔
Flag
CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table
Selects either the "Inactive" ([CFE_TBL_BufferSelect_INACTIVE](#)) buffer or the "Active" ([CFE_TBL_BufferSelect_ACTIVE](#))
buffer to be validated
Definition at line 79 of file default_cfe_tbl_msgdefs.h.

11.280.2.2 TableName char CFE_TBL_ValidateCmd_Payload::TableName [[CFE_MISSION_TBL_MAX_FULL_NAME_LEN](#)]
Full Name of Table to be validated.
ASCII string containing full table name identifier of table to be validated
Definition at line 85 of file default_cfe_tbl_msgdefs.h.
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h

11.281 CFE_TIME_AddAdjustCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_TimeCmd_Payload_t](#) Payload
Command payload.

11.281.1 Detailed Description

Definition at line 146 of file default_cfe_time_msgstruct.h.

11.281.2 Field Documentation

11.281.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_AddAdjustCmd::CommandHeader

Command header.

Definition at line 148 of file default_cfe_time_msgstruct.h.

11.281.2.2 Payload [CFE_TIME_TimeCmd_Payload_t](#) CFE_TIME_AddAdjustCmd::Payload

Command payload.

Definition at line 149 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.282 CFE_TIME_AddDelayCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_TimeCmd_Payload_t](#) Payload
Command payload.

11.282.1 Detailed Description

Definition at line 122 of file default_cfe_time_msgstruct.h.

11.282.2 Field Documentation

11.282.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_AddDelayCmd::CommandHeader

Command header.

Definition at line 124 of file default_cfe_time_msgstruct.h.

11.282.2.2 Payload [CFE_TIME_TimeCmd_Payload_t](#) CFE_TIME_AddDelayCmd::Payload
Command payload.

Definition at line 125 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.283 CFE_TIME_AddOneHzAdjustmentCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_OneHzAdjustmentCmd_Payload_t](#) Payload
Command payload.

11.283.1 Detailed Description

Definition at line 169 of file default_cfe_time_msgstruct.h.

11.283.2 Field Documentation

11.283.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_AddOneHzAdjustmentCmd::CommandHeader
Command header.

Definition at line 171 of file default_cfe_time_msgstruct.h.

11.283.2.2 Payload [CFE_TIME_OneHzAdjustmentCmd_Payload_t](#) CFE_TIME_AddOneHzAdjustmentCmd::Payload
Command payload.

Definition at line 172 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.284 CFE_TIME_DiagnosticTlm Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_TIME_DiagnosticTlm_Payload_t](#) Payload
Telemetry payload.

11.284.1 Detailed Description

Definition at line 198 of file default_cfe_time_msgstruct.h.

11.284.2 Field Documentation

11.284.2.1 Payload [CFE_TIME_DiagnosticTlm_Payload_t](#) CFE_TIME_DiagnosticTlm::Payload
Telemetry payload.

Definition at line 201 of file default_cfe_time_msgstruct.h.

11.284.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CFE_TIME_DiagnosticTlm::TelemetryHeader
Telemetry header.

Definition at line 200 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.285 CFE_TIME_DiagnosticTlm_Payload Struct Reference

```
#include <default_cfe_time_msgdefs.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.
- [CFE_TIME_ClockState_Enum_t 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)

11.285.1 Detailed Description

Name Time Services Diagnostics Packet

Definition at line 190 of file default_cfe_time_msgdefs.h.

11.285.2 Field Documentation

11.285.2.1 AtToneDelay [CFE_TIME_SysTime_t](#) CFE_TIME_DiagnosticTlm_Payload::AtToneDelay
Adjustment for slow tone detection.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLlatentS, \$sc_\$cpu_TIME_DLlatentSs

Definition at line 199 of file default_cfe_time_msgdefs.h.

11.285.2.2 AtToneLatch [CFE_TIME_SysTime_t](#) CFE_TIME_DiagnosticTlm_Payload::AtToneLatch
Local clock latched at time of tone.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTVlids, \$sc_\$cpu_TIME_DTVlidsS

Definition at line 201 of file default_cfe_time_msgdefs.h.

11.285.2.3 AtToneLeapSeconds [int16](#) CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds
Leap Seconds at time of tone.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLapS

Definition at line 204 of file default_cfe_time_msgdefs.h.

11.285.2.4 AtToneMET [CFE_TIME_SysTime_t](#) CFE_TIME_DiagnosticTlm_Payload::AtToneMET
MET at time of tone.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTMETS, \$sc_\$cpu_TIME_DTMETSs

Definition at line 195 of file default_cfe_time_msgdefs.h.

11.285.2.5 AtToneSTCF `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::AtToneSTCF`
STCF at time of tone.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DSTCFS`, `$sc_$cpu_TIME_DSTCFSS`

Definition at line 197 of file `default_cfe_time_msgdefs.h`.

11.285.2.6 ClockFlyState `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockFlyState`
Current fly-wheel state.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DFlywheel`

Definition at line 228 of file `default_cfe_time_msgdefs.h`.

11.285.2.7 ClockSetState `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSetState`
Time has been "set".

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DValid`

Definition at line 226 of file `default_cfe_time_msgdefs.h`.

11.285.2.8 ClockSignal `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSignal`
Primary vs redundant, etc.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DSignal`

Definition at line 232 of file `default_cfe_time_msgdefs.h`.

11.285.2.9 ClockSource `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSource`
Internal vs external, etc.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DSource`

Definition at line 230 of file `default_cfe_time_msgdefs.h`.

11.285.2.10 ClockStateAPI `CFE_TIME_ClockState_Enum_t` `CFE_TIME_DiagnosticTlm_Payload::ClockStateAPI`
Clock state as per API.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DAPIState`

Definition at line 206 of file `default_cfe_time_msgdefs.h`.

11.285.2.11 ClockStateFlags `uint16` `CFE_TIME_DiagnosticTlm_Payload::ClockStateFlags`
Clock State Flags.

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_DFlagAdj`, `$sc_$cpu_TIME_DFlag1Hzd`, `$sc_↔$cpu_TIME_DFlagClat`, `$sc_$cpu_TIME_DFlagSorC`, `$sc_$cpu_TIME_DFlagNIU`

Definition at line 242 of file `default_cfe_time_msgdefs.h`.

11.285.2.12 CurrentLatch `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::CurrentLatch`
Local clock latched just "now".

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DLLocalS`, `$sc_$cpu_TIME_DLLocalSs`

Definition at line 214 of file `default_cfe_time_msgdefs.h`.

11.285.2.13 CurrentMET `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::CurrentMET`
MET at this instant.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DMETS`, `$sc_$cpu_TIME_DMETSS`

Definition at line 216 of file `default_cfe_time_msgdefs.h`.

11.285.2.14 CurrentTAI `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::CurrentTAI`
TAI at this instant.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTAIS`, `$sc_$cpu_TIME_DTAISS`

Definition at line 218 of file `default_cfe_time_msgdefs.h`.

11.285.2.15 CurrentUTC `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::CurrentUTC`
UTC at this instant.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DUTCS`, `$sc_$cpu_TIME_DUTCSS`

Definition at line 220 of file `default_cfe_time_msgdefs.h`.

11.285.2.16 DataStoreStatus `uint32` `CFE_TIME_DiagnosticTlm_Payload::DataStoreStatus`
Data Store status (preserved across processor reset)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DataStStat`

Definition at line 332 of file `default_cfe_time_msgdefs.h`.

11.285.2.17 DelayDirection `int16` `CFE_TIME_DiagnosticTlm_Payload::DelayDirection`
Client latency adjustment direction.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DLlatentDir`

Definition at line 252 of file `default_cfe_time_msgdefs.h`.

11.285.2.18 Forced2Fly `int16` `CFE_TIME_DiagnosticTlm_Payload::Forced2Fly`
Commanded into fly-wheel.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DCMD2Fly`

Definition at line 236 of file `default_cfe_time_msgdefs.h`.

11.285.2.19 LocalIntCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::LocalIntCounter`
Local 1Hz ISR execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzISRCNT

Definition at line 290 of file default_cfe_time_msgdefs.h.

11.285.2.20 LocalTaskCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::LocalTaskCounter`
Local 1Hz task execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzTaskCNT

Definition at line 292 of file default_cfe_time_msgdefs.h.

11.285.2.21 MaxElapsed `uint32 CFE_TIME_DiagnosticTlm_Payload::MaxElapsed`
Max tone signal / data pkt arrival window (Sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DMaxWindow

Definition at line 312 of file default_cfe_time_msgdefs.h.

11.285.2.22 MaxLocalClock `CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload::MaxLocalClock`
Max local clock value before rollover.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DWrapS, \$sc_\$cpu_TIME_DWrapSs

Definition at line 318 of file default_cfe_time_msgdefs.h.

11.285.2.23 MinElapsed `uint32 CFE_TIME_DiagnosticTlm_Payload::MinElapsed`
Min tone signal / data pkt arrival window (Sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DMinWindow

Definition at line 310 of file default_cfe_time_msgdefs.h.

11.285.2.24 OneHzAdjust `CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload::OneHzAdjust`
Current 1Hz STCF adjustment.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzAdjS, \$sc_\$cpu_TIME_D1HzAdjSs

Definition at line 260 of file default_cfe_time_msgdefs.h.

11.285.2.25 OneHzDirection `int16 CFE_TIME_DiagnosticTlm_Payload::OneHzDirection`
1Hz STCF adjustment direction

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzAdjDir

Definition at line 250 of file default_cfe_time_msgdefs.h.

11.285.2.26 OneTimeAdjust `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::OneTimeAdjust`
Previous one-time STCF adjustment.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DAdjustS, \$sc_\$cpu_TIME_DAdjustSs

Definition at line 258 of file default_cfe_time_msgdefs.h.

11.285.2.27 OneTimeDirection `int16` `CFE_TIME_DiagnosticTlm_Payload::OneTimeDirection`
One time STCF adjustment direction (Add = 1, Sub = 2)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DAdjustDir

Definition at line 248 of file default_cfe_time_msgdefs.h.

11.285.2.28 ServerFlyState `int16` `CFE_TIME_DiagnosticTlm_Payload::ServerFlyState`
Used by clients only.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DSrvFly

Definition at line 234 of file default_cfe_time_msgdefs.h.

11.285.2.29 TimeSinceTone `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::TimeSinceTone`
Time elapsed since the tone.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DElapsedS, \$sc_\$cpu_TIME_DElapsedSs

Definition at line 212 of file default_cfe_time_msgdefs.h.

11.285.2.30 ToneDataCounter `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneDataCounter`
Time at the tone data SB message count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTatTCNT

Definition at line 280 of file default_cfe_time_msgdefs.h.

11.285.2.31 ToneDataLatch `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::ToneDataLatch`
Local Clock latched at arrival of tone data.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTDS, \$sc_\$cpu_TIME_DTDSs

Definition at line 268 of file default_cfe_time_msgdefs.h.

11.285.2.32 ToneIntCounter `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneIntCounter`
Tone signal ISR execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTsISRCNT

Definition at line 282 of file default_cfe_time_msgdefs.h.

11.285.2.33 ToneIntErrorCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneIntErrorCounter`
Tone signal ISR error count.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTsISRERR`

Definition at line 284 of file default_cfe_time_msgdefs.h.

11.285.2.34 ToneMatchCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneMatchCounter`
Tone signal / data verification count.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DVerifyCNT`

Definition at line 274 of file default_cfe_time_msgdefs.h.

11.285.2.35 ToneMatchErrorCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneMatchErrorCounter`
Tone signal / data verification error count.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DVerifyER`

Definition at line 276 of file default_cfe_time_msgdefs.h.

11.285.2.36 ToneOverLimit `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneOverLimit`
Max between tone signal interrupts.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DMaxSs`

Definition at line 324 of file default_cfe_time_msgdefs.h.

11.285.2.37 ToneSignalCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneSignalCounter`
Tone signal detected SB message count.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTSDetCNT`

Definition at line 278 of file default_cfe_time_msgdefs.h.

11.285.2.38 ToneSignalLatch `CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload::ToneSignalLatch`
Local Clock latched at most recent tone signal.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTTS, $sc_$cpu_TIME_DTTSS`

Definition at line 266 of file default_cfe_time_msgdefs.h.

11.285.2.39 ToneTaskCounter `uint32 CFE_TIME_DiagnosticTlm_Payload::ToneTaskCounter`
Tone task execution count.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTsTaskCNT`

Definition at line 286 of file default_cfe_time_msgdefs.h.

11.285.2.40 ToneUnderLimit `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneUnderLimit`
Min between tone signal interrupts.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DMinSs

Definition at line 326 of file default_cfe_time_msgdefs.h.

11.285.2.41 VersionCounter `uint32` `CFE_TIME_DiagnosticTlm_Payload::VersionCounter`
Count of mods to time at tone reference data (version)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DVersionCNT

Definition at line 288 of file default_cfe_time_msgdefs.h.

11.285.2.42 VirtualMET `uint32` `CFE_TIME_DiagnosticTlm_Payload::VirtualMET`
Software MET.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLLogicalMET

Definition at line 298 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.286 CFE_TIME_FakeToneCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t CommandHeader**
Command header.

11.286.1 Detailed Description

Definition at line 71 of file default_cfe_time_msgstruct.h.

11.286.2 Field Documentation

11.286.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TIME_FakeToneCmd::CommandHeader`
Command header.
Definition at line 73 of file default_cfe_time_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.287 CFE_TIME_HousekeepingTlm Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_TIME_HousekeepingTlm_Payload_t](#) Payload
Telemetry payload.

11.287.1 Detailed Description

Definition at line 192 of file `default_cfe_time_msgstruct.h`.

11.287.2 Field Documentation**11.287.2.1 Payload** [CFE_TIME_HousekeepingTlm_Payload_t](#) CFE_TIME_HousekeepingTlm::Payload
Telemetry payload.

Definition at line 195 of file `default_cfe_time_msgstruct.h`.

11.287.2.2 TelemetryHeader [CFE_MSG_TelemetryHeader_t](#) CFE_TIME_HousekeepingTlm::TelemetryHeader
Telemetry header.

Definition at line 194 of file `default_cfe_time_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.288 CFE_TIME_HousekeepingTlm_Payload Struct Reference

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- [uint8 CommandCounter](#)
Time Command Execution Counter.
- [uint8 CommandErrorCounter](#)
Time Command Error Counter.
- [uint16 ClockStateFlags](#)
State Flags.
- [CFE_TIME_ClockState_Enum_t](#) 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)

11.288.1 Detailed Description

Name Time Services Housekeeping Packet

Definition at line 127 of file default_cfe_time_msgdefs.h.

11.288.2 Field Documentation

11.288.2.1 ClockStateAPI `CFE_TIME_ClockState_Enum_t` `CFE_TIME_HousekeepingTlm_Payload::ClockState`↔
API
API State.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DAPIS

Definition at line 142 of file default_cfe_time_msgdefs.h.

11.288.2.2 ClockStateFlags `uint16` `CFE_TIME_HousekeepingTlm_Payload::ClockStateFlags`
State Flags.

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 140 of file default_cfe_time_msgdefs.h.

11.288.2.3 CommandCounter `uint8` `CFE_TIME_HousekeepingTlm_Payload::CommandCounter`
Time Command Execution Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_CMDPC

Definition at line 132 of file default_cfe_time_msgdefs.h.

11.288.2.4 CommandErrorCounter `uint8` `CFE_TIME_HousekeepingTlm_Payload::CommandErrorCounter`
Time Command Error Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_CMDEC

Definition at line 134 of file default_cfe_time_msgdefs.h.

11.288.2.5 LeapSeconds `int16` `CFE_TIME_HousekeepingTlm_Payload::LeapSeconds`
Current Leaps Seconds.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_LeapSecs`

Definition at line 148 of file `default_cfe_time_msgdefs.h`.

11.288.2.6 Seconds1HzAdj `uint32` `CFE_TIME_HousekeepingTlm_Payload::Seconds1HzAdj`
Current 1 Hz SCTF adjustment (seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_1HzAdjSecs`

Definition at line 168 of file `default_cfe_time_msgdefs.h`.

11.288.2.7 SecondsDelay `uint32` `CFE_TIME_HousekeepingTlm_Payload::SecondsDelay`
Current 1 Hz SCTF Delay (seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_1HzAdjSecs`

Definition at line 178 of file `default_cfe_time_msgdefs.h`.

11.288.2.8 SecondsMET `uint32` `CFE_TIME_HousekeepingTlm_Payload::SecondsMET`
Current MET (seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_METSecs`

Definition at line 154 of file `default_cfe_time_msgdefs.h`.

11.288.2.9 SecondsSTCF `uint32` `CFE_TIME_HousekeepingTlm_Payload::SecondsSTCF`
Current STCF (seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_STCFSecs`

Definition at line 159 of file `default_cfe_time_msgdefs.h`.

11.288.2.10 Subsecs1HzAdj `uint32` `CFE_TIME_HousekeepingTlm_Payload::Subsecs1HzAdj`
Current 1 Hz SCTF adjustment (sub-seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_1HzAdjSSecs`

Definition at line 170 of file `default_cfe_time_msgdefs.h`.

11.288.2.11 SubsecsDelay `uint32` `CFE_TIME_HousekeepingTlm_Payload::SubsecsDelay`
Current 1 Hz SCTF Delay (sub-seconds)

Telemetry Mnemonic(s) `$sc_$cpu_TIME_1HzAdjSSecs`

Definition at line 180 of file `default_cfe_time_msgdefs.h`.

11.288.2.12 SubsecsMET `uint32` `CFE_TIME_HousekeepingTlm_Payload::SubsecsMET`
Current MET (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_METSubsecs

Definition at line 156 of file default_cfe_time_msgdefs.h.

11.288.2.13 SubsecsSTCF `uint32` `CFE_TIME_HousekeepingTlm_Payload::SubsecsSTCF`
Current STCF (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_STCFSubsecs

Definition at line 161 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.289 CFE_TIME_LeapsCmd_Payload Struct Reference

Set leap seconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `int16 LeapSeconds`

11.289.1 Detailed Description

Set leap seconds command payload.

Definition at line 56 of file default_cfe_time_msgdefs.h.

11.289.2 Field Documentation

11.289.2.1 LeapSeconds `int16` `CFE_TIME_LeapsCmd_Payload::LeapSeconds`

Definition at line 58 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.290 CFE_TIME_NoopCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

Command header.

11.290.1 Detailed Description

Definition at line 46 of file default_cfe_time_msgstruct.h.

11.290.2 Field Documentation

11.290.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_TIME_NoopCmd::CommandHeader`
Command header.

Definition at line 50 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/time/config/default_cfe_time_msgstruct.h](#)

11.291 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference

Generic seconds, subseconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `uint32 Seconds`
- `uint32 Subseconds`

11.291.1 Detailed Description

Generic seconds, subseconds command payload.

Definition at line 105 of file default_cfe_time_msgdefs.h.

11.291.2 Field Documentation

11.291.2.1 Seconds `uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Seconds`

Definition at line 107 of file default_cfe_time_msgdefs.h.

11.291.2.2 Subseconds `uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Subseconds`

Definition at line 108 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/time/config/default_cfe_time_msgdefs.h](#)

11.292 CFE_TIME_OneHzCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- **CFE_MSG_CommandHeader_t CommandHeader**
Command header.

11.292.1 Detailed Description

Definition at line 61 of file default_cfe_time_msgstruct.h.

11.292.2 Field Documentation

11.292.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TIME_OneHzCmd::CommandHeader`
Command header.

Definition at line 63 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.293 CFE_TIME_ResetCountersCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.293.1 Detailed Description

Definition at line 51 of file default_cfe_time_msgstruct.h.

11.293.2 Field Documentation

11.293.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TIME_ResetCountersCmd::CommandHeader`
Command header.

Definition at line 53 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.294 CFE_TIME_SendDiagnosticCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.294.1 Detailed Description

Definition at line 56 of file default_cfe_time_msgstruct.h.

11.294.2 Field Documentation

11.294.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TIME_SendDiagnosticCmd::CommandHeader`
Command header.

Definition at line 58 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.295 CFE_TIME_SendHkCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

11.295.1 Detailed Description

Definition at line 76 of file default_cfe_time_msgstruct.h.

11.295.2 Field Documentation

11.295.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SendHkCmd::CommandHeader

Command header.

Definition at line 78 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.296 CFE_TIME_SetLeapSecondsCmd Struct Reference

Set leap seconds command.

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

- [CFE_TIME_LeapsCmd_Payload_t](#) Payload

Command payload.

11.296.1 Detailed Description

Set leap seconds command.

Definition at line 84 of file default_cfe_time_msgstruct.h.

11.296.2 Field Documentation

11.296.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetLeapSecondsCmd::CommandHeader

Command header.

Definition at line 86 of file default_cfe_time_msgstruct.h.

11.296.2.2 Payload [CFE_TIME_LeapsCmd_Payload_t](#) CFE_TIME_SetLeapSecondsCmd::Payload

Command payload.

Definition at line 87 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.297 CFE_TIME_SetMETCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_TimeCmd_Payload_t](#) Payload
Command payload.

11.297.1 Detailed Description

Definition at line 134 of file default_cfe_time_msgstruct.h.

11.297.2 Field Documentation

11.297.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetMETCmd::CommandHeader

Command header.

Definition at line 136 of file default_cfe_time_msgstruct.h.

11.297.2.2 Payload [CFE_TIME_TimeCmd_Payload_t](#) CFE_TIME_SetMETCmd::Payload

Command payload.

Definition at line 137 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.298 CFE_TIME_SetSignalCmd Struct Reference

Set tone signal source command.

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_SignalCmd_Payload_t](#) Payload
Command payload.

11.298.1 Detailed Description

Set tone signal source command.

Definition at line 111 of file default_cfe_time_msgstruct.h.

11.298.2 Field Documentation

11.298.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetSignalCmd::CommandHeader
Command header.
Definition at line 113 of file default_cfe_time_msgstruct.h.

11.298.2.2 Payload [CFE_TIME_SignalCmd_Payload_t](#) CFE_TIME_SetSignalCmd::Payload
Command payload.
Definition at line 114 of file default_cfe_time_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default_cfe_time_msgstruct.h](#)

11.299 CFE_TIME_SetSourceCmd Struct Reference

Set time data source command.

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_SourceCmd_Payload_t](#) Payload
Command payload.

11.299.1 Detailed Description

Set time data source command.

Definition at line 102 of file default_cfe_time_msgstruct.h.

11.299.2 Field Documentation

11.299.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetSourceCmd::CommandHeader
Command header.
Definition at line 104 of file default_cfe_time_msgstruct.h.

11.299.2.2 Payload [CFE_TIME_SourceCmd_Payload_t](#) CFE_TIME_SetSourceCmd::Payload
Command payload.
Definition at line 105 of file default_cfe_time_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default_cfe_time_msgstruct.h](#)

11.300 CFE_TIME_SetStateCmd Struct Reference

Set clock state command.

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_StateCmd_Payload_t](#) Payload
Command payload.

11.300.1 Detailed Description

Set clock state command.

Definition at line 93 of file default_cfe_time_msgstruct.h.

11.300.2 Field Documentation

11.300.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetStateCmd::CommandHeader

Command header.

Definition at line 95 of file default_cfe_time_msgstruct.h.

11.300.2.2 Payload [CFE_TIME_StateCmd_Payload_t](#) CFE_TIME_SetStateCmd::Payload

Command payload.

Definition at line 96 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default_cfe_time_msgstruct.h](#)

11.301 CFE_TIME_SetSTCFCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_TimeCmd_Payload_t](#) Payload
Command payload.

11.301.1 Detailed Description

Definition at line 140 of file default_cfe_time_msgstruct.h.

11.301.2 Field Documentation

11.301.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_SetSTCFCmd::CommandHeader

Command header.

Definition at line 142 of file default_cfe_time_msgstruct.h.

11.301.2.2 Payload `CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_SetSTCFCmd::Payload`
Command payload.

Definition at line 143 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.302 CFE_TIME_SetTimeCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_TimeCmd_Payload_t Payload`
Command payload.

11.302.1 Detailed Description

Definition at line 158 of file default_cfe_time_msgstruct.h.

11.302.2 Field Documentation

11.302.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TIME_SetTimeCmd::CommandHeader`
Command header.

Definition at line 160 of file default_cfe_time_msgstruct.h.

11.302.2.2 Payload `CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_SetTimeCmd::Payload`
Command payload.

Definition at line 161 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.303 CFE_TIME_SignalCmd_Payload Struct Reference

Set tone signal source command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `int16 ToneSource`
CFE_TIME_ToneSignalSelect_PRIMARY=Primary Source, CFE_TIME_ToneSignalSelect_REDUNDANT=Redundant Source

11.303.1 Detailed Description

Set tone signal source command payload.

Definition at line 86 of file default_cfe_time_msgdefs.h.

11.303.2 Field Documentation

11.303.2.1 ToneSource `int16 CFE_TIME_SignalCmd_Payload::ToneSource`
`CFE_TIME_ToneSignalSelect_PRIMARY`=Primary Source, `CFE_TIME_ToneSignalSelect_REDUNDANT`=Redundant Source

Selects either the "Primary" or "Redundant" tone signal source

Definition at line 88 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

11.304 CFE_TIME_SourceCmd_Payload Struct Reference

Set time data source command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `int16 TimeSource`
`CFE_TIME_SourceSelect_INTERNAL`=Internal Source, `CFE_TIME_SourceSelect_EXTERNAL`=External Source

11.304.1 Detailed Description

Set time data source command payload.

Definition at line 76 of file `default_cfe_time_msgdefs.h`.

11.304.2 Field Documentation

11.304.2.1 TimeSource `int16 CFE_TIME_SourceCmd_Payload::TimeSource`
`CFE_TIME_SourceSelect_INTERNAL`=Internal Source, `CFE_TIME_SourceSelect_EXTERNAL`=External Source

Selects either the "Internal" and "External" clock source

Definition at line 78 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

11.305 CFE_TIME_StateCmd_Payload Struct Reference

Set clock state command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `CFE_TIME_ClockState_Enum_t 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

11.305.1 Detailed Description

Set clock state command payload.

Definition at line 64 of file default_cfe_time_msgdefs.h.

11.305.2 Field Documentation

11.305.2.1 ClockState `CFE_TIME_ClockState_Enum_t CFE_TIME_StateCmd_Payload::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

Selects the current clock state

Definition at line 66 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.306 CFE_TIME_SubAdjustCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_TimeCmd_Payload_t Payload`
Command payload.

11.306.1 Detailed Description

Definition at line 152 of file default_cfe_time_msgstruct.h.

11.306.2 Field Documentation

11.306.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_TIME_SubAdjustCmd::CommandHeader`
Command header.
Definition at line 154 of file default_cfe_time_msgstruct.h.

11.306.2.2 Payload `CFE_TIME_TimeCmd_Payload_t CFE_TIME_SubAdjustCmd::Payload`
Command payload.
Definition at line 155 of file default_cfe_time_msgstruct.h.
The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.307 CFE_TIME_SubDelayCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_TimeCmd_Payload_t](#) Payload
Command payload.

11.307.1 Detailed Description

Definition at line 128 of file default_cfe_time_msgstruct.h.

11.307.2 Field Documentation**11.307.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_TIME_SubDelayCmd::CommandHeader
Command header.

Definition at line 130 of file default_cfe_time_msgstruct.h.

11.307.2.2 Payload [CFE_TIME_TimeCmd_Payload_t](#) CFE_TIME_SubDelayCmd::Payload
Command payload.

Definition at line 131 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.308 CFE_TIME_SubOneHzAdjustmentCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_OneHzAdjustmentCmd_Payload_t](#) Payload
Command payload.

11.308.1 Detailed Description

Definition at line 175 of file default_cfe_time_msgstruct.h.

11.308.2 Field Documentation**11.308.2.1 CommandHeader** [CFE_MSG_CommandHeader_t](#) CFE_TIME_SubOneHzAdjustmentCmd::CommandHeader
Command header.

Definition at line 177 of file default_cfe_time_msgstruct.h.

11.308.2.2 Payload `CFE_TIME_OneHzAdjustmentCmd_Payload_t` `CFE_TIME_SubOneHzAdjustmentCmd::Payload`
Command payload.

Definition at line 178 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

11.309 CFE_TIME_SysTime Struct Reference

Data structure used to hold system time values.

```
#include <default_cfe_time_extern_typedefs.h>
```

Data Fields

- `uint32 Seconds`
Number of seconds since epoch.
- `uint32 Subseconds`
Number of subseconds since epoch (LSB = 2^{-32} seconds)

11.309.1 Detailed Description

Data structure used to hold system time values.

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 41 of file default_cfe_time_extern_typedefs.h.

11.309.2 Field Documentation

11.309.2.1 Seconds `uint32 CFE_TIME_SysTime::Seconds`

Number of seconds since epoch.

Definition at line 43 of file default_cfe_time_extern_typedefs.h.

11.309.2.2 Subseconds `uint32 CFE_TIME_SysTime::Subseconds`

Number of subseconds since epoch (LSB = 2^{-32} seconds)

Definition at line 44 of file default_cfe_time_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_extern_typedefs.h`

11.310 CFE_TIME_TimeCmd_Payload Struct Reference

Generic seconds, microseconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

Data Fields

- `uint32 Seconds`
- `uint32 MicroSeconds`

11.310.1 Detailed Description

Generic seconds, microseconds command payload.

Definition at line 96 of file default_cfe_time_msgdefs.h.

11.310.2 Field Documentation

11.310.2.1 MicroSeconds `uint32` CFE_TIME_TimeCmd_Payload::MicroSeconds

Definition at line 99 of file default_cfe_time_msgdefs.h.

11.310.2.2 Seconds `uint32` CFE_TIME_TimeCmd_Payload::Seconds

Definition at line 98 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.311 CFE_TIME_ToneDataCmd Struct Reference

Time at tone data command.

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TIME_ToneDataCmd_Payload_t](#) Payload
Command payload.

11.311.1 Detailed Description

Time at tone data command.

Definition at line 184 of file default_cfe_time_msgstruct.h.

11.311.2 Field Documentation

11.311.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_ToneDataCmd::CommandHeader

Command header.

Definition at line 186 of file default_cfe_time_msgstruct.h.

11.311.2.2 Payload [CFE_TIME_ToneDataCmd_Payload_t](#) CFE_TIME_ToneDataCmd::Payload

Command payload.

Definition at line 187 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.312 CFE_TIME_ToneDataCmd_Payload Struct Reference

Time at tone data command payload.

```
#include <default_cfe_time_msgdefs.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.
- [CFE_TIME_ClockState_Enum_t AtToneState](#)
Clock state at time of tone.

11.312.1 Detailed Description

Time at tone data command payload.

Definition at line 114 of file default_cfe_time_msgdefs.h.

11.312.2 Field Documentation

11.312.2.1 AtToneLeapSeconds [int16 CFE_TIME_ToneDataCmd_Payload::AtToneLeapSeconds](#)

Leap Seconds at time of tone.

Definition at line 118 of file default_cfe_time_msgdefs.h.

11.312.2.2 AtToneMET [CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneMET](#)

MET at time of tone.

Definition at line 116 of file default_cfe_time_msgdefs.h.

11.312.2.3 AtToneState [CFE_TIME_ClockState_Enum_t CFE_TIME_ToneDataCmd_Payload::AtToneState](#)

Clock state at time of tone.

Definition at line 119 of file default_cfe_time_msgdefs.h.

11.312.2.4 AtToneSTCF [CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneSTCF](#)

STCF at time of tone.

Definition at line 117 of file default_cfe_time_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgdefs.h

11.313 CFE_TIME_ToneSignalCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader

Command header.

11.313.1 Detailed Description

Definition at line 66 of file default_cfe_time_msgstruct.h.

11.313.2 Field Documentation**11.313.2.1 CommandHeader [CFE_MSG_CommandHeader_t](#) CFE_TIME_ToneSignalCmd::CommandHeader**

Command header.

Definition at line 68 of file default_cfe_time_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default_cfe_time_msgstruct.h

11.314 OS_bin_sem_prop_t Struct Reference

OSAL binary semaphore properties.

```
#include <osapi-binsem.h>
```

Data Fields

- char [name](#) [[OS_MAX_API_NAME](#)]
- [osal_id_t](#) creator
- [int32](#) value

11.314.1 Detailed Description

OSAL binary semaphore properties.

Definition at line 39 of file osapi-binsem.h.

11.314.2 Field Documentation**11.314.2.1 creator [osal_id_t](#) OS_bin_sem_prop_t::creator**

Definition at line 42 of file osapi-binsem.h.

11.314.2.2 name [char](#) OS_bin_sem_prop_t::name[[OS_MAX_API_NAME](#)]

Definition at line 41 of file osapi-binsem.h.

11.314.2.3 value [int32](#) OS_bin_sem_prop_t::value

Definition at line 43 of file osapi-binsem.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-binsem.h](#)

11.315 OS_condvar_prop_t Struct Reference

OSAL condition variable properties.

```
#include <osapi-condvar.h>
```

Data Fields

- char `name` [`OS_MAX_API_NAME`]
- `osal_id_t creator`

11.315.1 Detailed Description

OSAL condition variable properties.

Definition at line 34 of file osapi-condvar.h.

11.315.2 Field Documentation

11.315.2.1 `creator` `osal_id_t` `OS_condvar_prop_t::creator`

Definition at line 37 of file osapi-condvar.h.

11.315.2.2 `name` char `OS_condvar_prop_t::name[OS_MAX_API_NAME]`

Definition at line 36 of file osapi-condvar.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-condvar.h`

11.316 OS_count_sem_prop_t Struct Reference

OSAL counting semaphore properties.

```
#include <osapi-countsem.h>
```

Data Fields

- char `name` [`OS_MAX_API_NAME`]
- `osal_id_t creator`
- `int32 value`

11.316.1 Detailed Description

OSAL counting semaphore properties.

Definition at line 32 of file osapi-countsem.h.

11.316.2 Field Documentation

11.316.2.1 `creator` `osal_id_t` `OS_count_sem_prop_t::creator`

Definition at line 35 of file osapi-countsem.h.

11.316.2.2 name char OS_count_sem_prop_t::name[OS_MAX_API_NAME]
Definition at line 34 of file osapi-countsem.h.

11.316.2.3 value int32 OS_count_sem_prop_t::value

Definition at line 36 of file osapi-countsem.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-countsem.h

11.317 os_dirent_t Struct Reference

Directory entry.

```
#include <osapi-dir.h>
```

Data Fields

- char **FileName** [OS_MAX_FILE_NAME]

11.317.1 Detailed Description

Directory entry.

Definition at line 32 of file osapi-dir.h.

11.317.2 Field Documentation

11.317.2.1 FileName char os_dirent_t::FileName[OS_MAX_FILE_NAME]

Definition at line 34 of file osapi-dir.h.

Referenced by CF_CFDP_ProcessPlaybackDirectory().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-dir.h

11.318 OS_FdSet Struct Reference

An abstract structure capable of holding several OSAL IDs.

```
#include <osapi-select.h>
```

Data Fields

- uint8 **object_ids** [(OS_MAX_NUM_OPEN_FILES+7)/8]

11.318.1 Detailed Description

An abstract structure capable of holding several OSAL IDs.

This is part of the select API and is manipulated using the related API calls. It should not be modified directly by applications.

Note: Math is to determine uint8 array size needed to represent single bit OS_MAX_NUM_OPEN_FILES objects, + 7 rounds up and 8 is the size of uint8.

See also

[OS_SelectFdZero\(\)](#), [OS_SelectFdAdd\(\)](#), [OS_SelectFdClear\(\)](#), [OS_SelectFdIsSet\(\)](#)

Definition at line 44 of file osapi-select.h.

11.318.2 Field Documentation

11.318.2.1 **object_ids** `uint8 OS_FdSet::object_ids[(OS_MAX_NUM_OPEN_FILES+7) / 8]`

Definition at line 46 of file osapi-select.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-select.h](#)

11.319 OS_file_prop_t Struct Reference

OSAL file properties.

```
#include <osapi-file.h>
```

Data Fields

- `char Path [OS_MAX_PATH_LEN]`
- `osal_id_t User`
- `uint8 IsValid`

11.319.1 Detailed Description

OSAL file properties.

Definition at line 49 of file osapi-file.h.

11.319.2 Field Documentation

11.319.2.1 **IsValid** `uint8 OS_file_prop_t::IsValid`

Definition at line 53 of file osapi-file.h.

11.319.2.2 **Path** `char OS_file_prop_t::Path[OS_MAX_PATH_LEN]`

Definition at line 51 of file osapi-file.h.

11.319.2.3 **User** `osal_id_t OS_file_prop_t::User`

Definition at line 52 of file osapi-file.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-file.h](#)

11.320 os_fsinfo_t Struct Reference

OSAL file system info.

```
#include <osapi-fsinfo.h>
```

Data Fields

- `uint32 MaxFds`
Total number of file descriptors.
- `uint32 FreeFds`

Total number that are free.

- **uint32 MaxVolumes**

Maximum number of volumes.

- **uint32 FreeVolumes**

Total number of volumes free.

11.320.1 Detailed Description

OSAL file system info.

Definition at line 35 of file osapi-filesystem.h.

11.320.2 Field Documentation

11.320.2.1 **FreeFds** `uint32 os_fsinfo_t::FreeFds`

Total number that are free.

Definition at line 38 of file osapi-filesystem.h.

11.320.2.2 **FreeVolumes** `uint32 os_fsinfo_t::FreeVolumes`

Total number of volumes free.

Definition at line 40 of file osapi-filesystem.h.

11.320.2.3 **MaxFds** `uint32 os_fsinfo_t::MaxFds`

Total number of file descriptors.

Definition at line 37 of file osapi-filesystem.h.

11.320.2.4 **MaxVolumes** `uint32 os_fsinfo_t::MaxVolumes`

Maximum number of volumes.

Definition at line 39 of file osapi-filesystem.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-filesystem.h`

11.321 os_fstat_t Struct Reference

File system status.

```
#include <osapi-file.h>
```

Data Fields

- **uint32 FileModeBits**
- **OS_time_t FileTime**
- **size_t FileSize**

11.321.1 Detailed Description

File system status.

Note

This used to be directly typedef'ed to the "struct stat" from the C library

Some C libraries (glibc in particular) actually define member names to reference into sub-structures, so attempting to reuse a name like "st_mtime" might not work.

Definition at line 64 of file osapi-file.h.

11.321.2 Field Documentation

11.321.2.1 FileModeBits `uint32` `os_fstat_t::FileModeBits`

Definition at line 66 of file osapi-file.h.

11.321.2.2 FileSize `size_t` `os_fstat_t::FileSize`

Definition at line 68 of file osapi-file.h.

11.321.2.3 FileTime `OS_time_t` `os_fstat_t::FileTime`

Definition at line 67 of file osapi-file.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-file.h`

11.322 OS_heap_prop_t Struct Reference

OSAL heap properties.

```
#include <osapi-heap.h>
```

Data Fields

- `size_t free_bytes`
- `osal_blockcount_t free_blocks`
- `size_t largest_free_block`

11.322.1 Detailed Description

OSAL heap properties.

See also

[OS_HeapGetInfo\(\)](#)

Definition at line 36 of file osapi-heap.h.

11.322.2 Field Documentation

11.322.2.1 free_blocks `osal_blockcount_t` `OS_heap_prop_t::free_blocks`

Definition at line 39 of file osapi-heap.h.

11.322.2.2 free_bytes size_t OS_heap_prop_t::free_bytes
Definition at line 38 of file osapi-heap.h.

11.322.2.3 largest_free_block size_t OS_heap_prop_t::largest_free_block
Definition at line 40 of file osapi-heap.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-heap.h](#)

11.323 OS_module_address_t Struct Reference

OSAL module address properties.

```
#include <osapi-module.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](#)

11.323.1 Detailed Description

OSAL module address properties.

Definition at line 78 of file osapi-module.h.

11.323.2 Field Documentation

11.323.2.1 bss_address [cpuaddr](#) OS_module_address_t::bss_address
Definition at line 86 of file osapi-module.h.

11.323.2.2 bss_size [cpuaddr](#) OS_module_address_t::bss_size
Definition at line 87 of file osapi-module.h.

11.323.2.3 code_address [cpuaddr](#) OS_module_address_t::code_address
Definition at line 82 of file osapi-module.h.

11.323.2.4 code_size [cpuaddr](#) OS_module_address_t::code_size
Definition at line 83 of file osapi-module.h.

11.323.2.5 data_address [cpuaddr](#) OS_module_address_t::data_address
Definition at line 84 of file osapi-module.h.

11.323.2.6 data_size `cpuaddr OS_module_address_t::data_size`
Definition at line 85 of file osapi-module.h.

11.323.2.7 flags `uint32 OS_module_address_t::flags`
Definition at line 81 of file osapi-module.h.

11.323.2.8 valid `uint32 OS_module_address_t::valid`
Definition at line 80 of file osapi-module.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-module.h`

11.324 OS_module_prop_t Struct Reference

OSAL module properties.

```
#include <osapi-module.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`

11.324.1 Detailed Description

OSAL module properties.

Definition at line 91 of file osapi-module.h.

11.324.2 Field Documentation

11.324.2.1 addr `OS_module_address_t OS_module_prop_t::addr`
Definition at line 97 of file osapi-module.h.

11.324.2.2 entry_point `cpuaddr OS_module_prop_t::entry_point`
Definition at line 93 of file osapi-module.h.

11.324.2.3 filename `char OS_module_prop_t::filename[OS_MAX_PATH_LEN]`
Definition at line 95 of file osapi-module.h.

11.324.2.4 host_module_id `cpuaddr OS_module_prop_t::host_module_id`
Definition at line 94 of file osapi-module.h.

11.324.2.5 name char OS_module_prop_t::name[OS_MAX_API_NAME]

Definition at line 96 of file osapi-module.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-module.h](#)

11.325 OS_mut_sem_prop_t Struct Reference

OSAL mutex properties.

```
#include <osapi-mutex.h>
```

Data Fields

- char [name](#) [OS_MAX_API_NAME]
- [osal_id_t creator](#)

11.325.1 Detailed Description

OSAL mutex properties.

Definition at line 32 of file osapi-mutex.h.

11.325.2 Field Documentation

11.325.2.1 creator [osal_id_t](#) OS_mut_sem_prop_t::creator

Definition at line 35 of file osapi-mutex.h.

11.325.2.2 name char OS_mut_sem_prop_t::name[OS_MAX_API_NAME]

Definition at line 34 of file osapi-mutex.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-mutex.h](#)

11.326 OS_queue_prop_t Struct Reference

OSAL queue properties.

```
#include <osapi-queue.h>
```

Data Fields

- char [name](#) [OS_MAX_API_NAME]
- [osal_id_t creator](#)

11.326.1 Detailed Description

OSAL queue properties.

Definition at line 32 of file osapi-queue.h.

11.326.2 Field Documentation

11.326.2.1 creator `osal_id_t OS_queue_prop_t::creator`
Definition at line 35 of file osapi-queue.h.

11.326.2.2 name `char OS_queue_prop_t::name[OS_MAX_API_NAME]`
Definition at line 34 of file osapi-queue.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-queue.h`

11.327 OS_SockAddr_t Struct Reference

Encapsulates a generic network address.

```
#include <osapi-sockets.h>
```

Data Fields

- `size_t ActualLength`
Length of the actual address data.
- `OS_SockAddrData_t AddrData`
Abstract Address data.

11.327.1 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 110 of file osapi-sockets.h.

11.327.2 Field Documentation

11.327.2.1 ActualLength `size_t OS_SockAddr_t::ActualLength`
Length of the actual address data.

Definition at line 112 of file osapi-sockets.h.

11.327.2.2 AddrData `OS_SockAddrData_t OS_SockAddr_t::AddrData`
Abstract Address data.

Definition at line 113 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-sockets.h`

11.328 OS_SockAddrData_t Union Reference

Storage buffer for generic network address.

```
#include <osapi-sockets.h>
```

Data Fields

- `uint8 Buffer [OS SOCKADDR_MAX_LEN]`
Ensures length of at least OS SOCKADDR_MAX_LEN.
- `uint32 AlignU32`
Ensures uint32 alignment.
- `void * AlignPtr`
Ensures pointer alignment.

11.328.1 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 96 of file osapi-sockets.h.

11.328.2 Field Documentation**11.328.2.1 AlignPtr `void* OS_SockAddrData_t::AlignPtr`**

Ensures pointer alignment.

Definition at line 100 of file osapi-sockets.h.

11.328.2.2 AlignU32 `uint32 OS_SockAddrData_t::AlignU32`

Ensures uint32 alignment.

Definition at line 99 of file osapi-sockets.h.

11.328.2.3 Buffer `uint8 OS_SockAddrData_t::Buffer[OS SOCKADDR_MAX_LEN]`

Ensures length of at least OS SOCKADDR_MAX_LEN.

Definition at line 98 of file osapi-sockets.h.

The documentation for this union was generated from the following file:

- `osal/src/os/inc/osapi-sockets.h`

11.329 OS_socket_prop_t Struct Reference

Encapsulates socket properties.

```
#include <osapi-sockets.h>
```

Data Fields

- `char name [OS_MAX_API_NAME]`
Name of the socket.
- `osal_id_t creator`
OSAL TaskID which opened the socket.

11.329.1 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 123 of file osapi-sockets.h.

11.329.2 Field Documentation

11.329.2.1 **creator** `osal_id_t OS_socket_prop_t::creator`

OSAL TaskID which opened the socket.

Definition at line 126 of file osapi-sockets.h.

11.329.2.2 **name** `char OS_socket_prop_t::name[OS_MAX_API_NAME]`

Name of the socket.

Definition at line 125 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-sockets.h`

11.330 OS_static_symbol_record_t Struct Reference

Associates a single symbol name with a memory address.

#include <osapi-module.h>

Data Fields

- `const char * Name`
- `void(* Address)(void)`
- `const char * Module`

11.330.1 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 113 of file osapi-module.h.

11.330.2 Field Documentation

11.330.2.1 **Address** `void(* OS_static_symbol_record_t::Address)(void)`

Definition at line 116 of file osapi-module.h.

11.330.2.2 **Module** `const char* OS_static_symbol_record_t::Module`

Definition at line 117 of file osapi-module.h.

11.330.2.3 **Name** `const char* OS_static_symbol_record_t::Name`

Definition at line 115 of file osapi-module.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-module.h`

11.331 OS_statvfs_t Struct Reference

```
#include <osapi-filesystems.h>
```

Data Fields

- size_t **block_size**
- osal_blockcount_t **total_blocks**
- osal_blockcount_t **blocks_free**

11.331.1 Detailed Description

Definition at line 49 of file osapi-filesystems.h.

11.331.2 Field Documentation

11.331.2.1 **block_size** size_t OS_statvfs_t::block_size

Block size of underlying FS

Definition at line 51 of file osapi-filesystems.h.

11.331.2.2 **blocks_free** osal_blockcount_t OS_statvfs_t::blocks_free

Available blocks in underlying FS

Definition at line 53 of file osapi-filesystems.h.

11.331.2.3 **total_blocks** osal_blockcount_t OS_statvfs_t::total_blocks

Total blocks in underlying FS

Definition at line 52 of file osapi-filesystems.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-filesystems.h](#)

11.332 OS_task_prop_t Struct Reference

OSAL task properties.

```
#include <osapi-task.h>
```

Data Fields

- char **name** [OS_MAX_API_NAME]
- osal_id_t **creator**
- size_t **stack_size**
- osal_priority_t **priority**

11.332.1 Detailed Description

OSAL task properties.

Definition at line 57 of file osapi-task.h.

11.332.2 Field Documentation

11.332.2.1 creator `osal_id_t OS_task_prop_t::creator`
Definition at line 60 of file osapi-task.h.

11.332.2.2 name `char OS_task_prop_t::name[OS_MAX_API_NAME]`
Definition at line 59 of file osapi-task.h.

11.332.2.3 priority `osal_priority_t OS_task_prop_t::priority`
Definition at line 62 of file osapi-task.h.

11.332.2.4 stack_size `size_t OS_task_prop_t::stack_size`
Definition at line 61 of file osapi-task.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-task.h](#)

11.333 OS_time_t Struct Reference

OSAL time interval structure.

```
#include <osapi-clock.h>
```

Data Fields

- [int64 ticks](#)

11.333.1 Detailed Description

OSAL time interval structure.

This is used to represent a basic time interval.

When used with OS_GetLocalTime/OS_SetLocalTime, this represents the interval from the OS's epoch point, typically 01 Jan 1970 00:00:00 UTC on systems that have a persistent real time clock (RTC), or the system boot time if there is no RTC available.

Applications should not directly access fields within this structure, as the definition may change in future versions of OSAL. Instead, applications should use the accessor/conversion methods defined below.

Definition at line 45 of file osapi-clock.h.

11.333.2 Field Documentation

11.333.2.1 ticks `int64 OS_time_t::ticks`

Ticks elapsed since reference point

Definition at line 47 of file osapi-clock.h.

Referenced by OS_TimeAdd(), OS_TimeAssembleFromMicroseconds(), OS_TimeAssembleFromMilliseconds(), OS_TimeAssembleFromNanoseconds(), OS_TimeAssembleFromSubseconds(), OS_TimeEqual(), OS_TimeGetFractionalPart(), OS_TimeGetSign(), OS_TimeGetTotalMicroseconds(), OS_TimeGetTotalMilliseconds(), OS_TimeGetTotalNanoseconds(), OS_TimeGetTotalSeconds(), and OS_TimeSubtract().

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-clock.h](#)

11.334 OS_timebase_prop_t Struct Reference

Time base properties.

```
#include <osapi-timebase.h>
```

Data Fields

- char `name` [OS_MAX_API_NAME]
- `osal_id_t` `creator`
- `uint32` `nominal_interval_time`
- `uint32` `freerun_time`
- `uint32` `accuracy`

11.334.1 Detailed Description

Time base properties.

Definition at line 37 of file osapi-timebase.h.

11.334.2 Field Documentation

11.334.2.1 `accuracy` `uint32` OS_timebase_prop_t::accuracy

Definition at line 43 of file osapi-timebase.h.

11.334.2.2 `creator` `osal_id_t` OS_timebase_prop_t::creator

Definition at line 40 of file osapi-timebase.h.

11.334.2.3 `freerun_time` `uint32` OS_timebase_prop_t::freerun_time

Definition at line 42 of file osapi-timebase.h.

11.334.2.4 `name` `char` OS_timebase_prop_t::name[OS_MAX_API_NAME]

Definition at line 39 of file osapi-timebase.h.

11.334.2.5 `nominal_interval_time` `uint32` OS_timebase_prop_t::nominal_interval_time

Definition at line 41 of file osapi-timebase.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-timebase.h](#)

11.335 OS_timer_prop_t Struct Reference

Timer properties.

```
#include <osapi-timer.h>
```

Data Fields

- char `name` [`OS_MAX_API_NAME`]
- `osal_id_t` `creator`
- `uint32` `start_time`
- `uint32` `interval_time`
- `uint32` `accuracy`

11.335.1 Detailed Description

Timer properties.

Definition at line 37 of file osapi-timer.h.

11.335.2 Field Documentation

11.335.2.1 `accuracy` `uint32` `OS_timer_prop_t::accuracy`

Definition at line 43 of file osapi-timer.h.

11.335.2.2 `creator` `osal_id_t` `OS_timer_prop_t::creator`

Definition at line 40 of file osapi-timer.h.

11.335.2.3 `interval_time` `uint32` `OS_timer_prop_t::interval_time`

Definition at line 42 of file osapi-timer.h.

11.335.2.4 `name` `char` `OS_timer_prop_t::name[OS_MAX_API_NAME]`

Definition at line 39 of file osapi-timer.h.

11.335.2.5 `start_time` `uint32` `OS_timer_prop_t::start_time`

Definition at line 41 of file osapi-timer.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-timer.h`

12 File Documentation

12.1 `apps/cf/config/default_cf_extern_typedefs.h` File Reference

```
#include "cf_mission_cfg.h"
```

Data Structures

- struct `CF_TxnFilenames`

Cache of source and destination filename.

Typedefs

- **typedef struct CF_TxnFilenames CF_TxnFilenames_t**
Cache of source and destination filename.
- **typedef uint32 CF_EntityId_t**
Entity id size.
- **typedef uint32 CF_TransactionSeq_t**
transaction sequence number size

Enumerations

- **enum CF_CFDP_Class_t { CF_CFDP_CLASS_1 = 0, CF_CFDP_CLASS_2 = 1 }**
Values for CFDP file transfer class.
- **enum CF_QueueIdx_t {**
CF_QueueIdx_PEND = 0, CF_QueueIdx_TXA = 1, CF_QueueIdx_TXW = 2, CF_QueueIdx_RX = 3,
CF_QueueIdx_HIST = 4, CF_QueueIdx_HIST_FREE = 5, CF_QueueIdx_FREE = 6, CF_QueueIdx_NUM = 7 }
CF queue identifiers.

12.1.1 Detailed Description

Declarations and prototypes for cf_extern_typedefs module

12.1.2 Typedef Documentation

12.1.2.1 CF_EntityId_t `typedef uint32 CF_EntityId_t`

Entity id size.

Description:

The maximum size of the entity id as expected for all CFDP packets. CF supports the spec's variable size of EID, where the actual size is selected at runtime, and therefore the size in CFDP PDUs may be smaller than the size specified here. This type only establishes the maximum size (and therefore maximum value) that an EID may be.

Note

This type is used in several CF commands, and so changing the size of this type will affect the following structs: CF_ConfigTable_t, configuration table - will change size of file CF_ConfigPacket_t, set config params command CF_TxFileCmd_t, transmit file command CF_PlaybackDirCmd_t, equivalent to above CF_Transaction_Payload_t, any command that selects a transaction based on EID

Limits

Must be one of uint8, uint16, uint32, uint64.

Definition at line 117 of file default_cf_extern_typedefs.h.

12.1.2.2 CF_TransactionSeq_t `typedef uint32 CF_TransactionSeq_t`

transaction sequence number size

Description:

The max size of the transaction sequence number as expected for all CFDP packets. CF supports the spec's variable size of TSN, where the actual size is selected at runtime, and therefore the size in CFDP PDUs may be smaller than the size specified here. This type only establishes the maximum size (and therefore maximum value) that a TSN may be.

Note

This type is used in several CF commands, and so changing the size of this type will affect the following structure: CF_Transaction_Payload_t, any command that selects a transaction based on TSN

Limits

Must be one of uint8, uint16, uint32, uint64.

Definition at line 136 of file default_cf_extern_typedefs.h.

12.1.2.3 CF_TxnFilenames_t `typedef struct CF_TxnFilenames CF_TxnFilenames_t`

Cache of source and destination filename.

This pairs a source and destination file name together to be retained for future reference in the transaction/history

12.1.3 Enumeration Type Documentation

12.1.3.1 CF_CFDP_Class_t `enum CF_CFDP_Class_t`

Values for CFDP file transfer class.

The CFDP specification prescribes two classes/modes of file transfer protocol operation - unacknowledged/simple or acknowledged/reliable.

Defined per section 7.1 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_CLASS_1	CFDP class 1 - Unreliable transfer.
CF_CFDP_CLASS_2	CFDP class 2 - Reliable transfer.

Definition at line 63 of file default_cf_extern_typedefs.h.

12.1.3.2 CF_QueueIdx_t `enum CF_QueueIdx_t`

CF queue identifiers.

Enumerator

CF_QueueIdx_PEND	first one on this list is active
CF_QueueIdx_TXA	
CF_QueueIdx_TXW	
CF_QueueIdx_RX	
CF_QueueIdx_HIST	
CF_QueueIdx_HIST_FREE	

Enumerator

CF_Queueldx_FREE	
CF_Queueldx_NUM	

Definition at line 72 of file default_cf_extern_typedefs.h.

12.2 apps/cf/config/default_cf_fcncodes.h File Reference

Enumerations

- enum **CF_CMDS** {

CF_NOOP_CC = 0, CF_RESET_CC = 1, CF_TX_FILE_CC = 2, CF_PLAYBACK_DIR_CC = 3,

CF_FREEZE_CC = 4, CF_THAW_CC = 5, CF_SUSPEND_CC = 6, CF_RESUME_CC = 7,

CF_CANCEL_CC = 8, CF_ABANDON_CC = 9, CF_SET_PARAM_CC = 10, CF_GET_PARAM_CC = 11,

CF_WRITE_QUEUE_CC = 15, CF_ENABLE_DEQUEUE_CC = 16, CF_DISABLE_DEQUEUE_CC = 17,

CF_ENABLE_DIR_POLLING_CC = 18,

CF_DISABLE_DIR_POLLING_CC = 19, CF_PURGE_QUEUE_CC = 21, CF_ENABLE_ENGINE_CC = 22,

CF_DISABLE_ENGINE_CC = 23,

CF_NUM_COMMANDS = 24 }

12.2.1 Detailed Description

Specification for the CFS CFDP (CF) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.3 apps/cf/config/default_cf_interface_cfg.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
```

Macros

- #define **CF_NUM_CHANNELS** (2)

Number of channels.
- #define **CF_NAK_MAX_SEGMENTS** (58)

Max NAK segments supported in a NAK PDU.
- #define **CF_MAX_POLLING_DIR_PER_CHAN** (5)

Max number of polling directories per channel.
- #define **CF_MAX_PDU_SIZE** (512)

Max PDU size.
- #define **CF_FILENAME_MAX_NAME** CFE_MISSION_MAX_FILE_LEN

Maximum file name length.
- #define **CF_FILENAME_MAX_LEN** CFE_MISSION_MAX_PATH_LEN

Max filename and path length.
- #define **CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES** 0

Number of trailing bytes to add to CFDP PDU.

12.3.1 Detailed Description

CFS CFDP (CF) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.4 apps/cf/config/default_cf_internal_cfg.h File Reference

Macros

- `#define CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION`
RX chunks per transaction (per channel)
- `#define CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION`
TX chunks per transaction (per channel)
- `#define CF_PIPE_DEPTH (32)`
Application Pipe Depth.
- `#define CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN (10)`
Number of max commanded playback files per chan.
- `#define CF_MAX_SIMULTANEOUS_RX (5)`
Max number of simultaneous file receives.
- `#define CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN (2)`
Max number of commanded playback directories per channel.
- `#define CF_NUM_HISTORIES_PER_CHANNEL (256)`
Number of histories per channel.
- `#define CF_NUM_TRANSACTIONS_PER_PLAYBACK (5)`
Number of transactions per playback directory.
- `#define CF_CONFIG_TABLE_NAME ("config_table")`
Name of the CF Configuration Table.
- `#define CF_CONFIG_TABLE_FILENAME ("/cf/cf_def_config.tbl")`
CF Configuration Table Filename.
- `#define CF_R2_CRC_CHUNK_SIZE (1024)`
R2 CRC calc chunk size.
- `#define CF_RCVMSG_TIMEOUT (100)`
Number of milliseconds to wait for a SB message.
- `#define CF_STARTUP_SEM_MAX_RETRIES 25`
Limits the number of retries to obtain the CF throttle sem.
- `#define CF_STARTUP_SEM_TASK_DELAY 100`
Number of milliseconds to wait if CF throttle sem is not available.

12.4.1 Detailed Description

CFS CFDP (CF) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.5 apps/cf/config/default_cf_mission_cfg.h File Reference

```
#include "cf_interface_cfg.h"
```

Macros

- #define CF_FILENAME_MAX_PATH (CFE_MISSION_MAX_PATH_LEN - CFE_MISSION_MAX_FILE_LEN)
Maximum file path (not including file name)

12.5.1 Detailed Description

CFS CFDP (CF) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.5.2 Macro Definition Documentation

12.5.2.1 CF_FILENAME_MAX_PATH #define CF_FILENAME_MAX_PATH (CFE_MISSION_MAX_PATH_LEN - CFE_MISSION_MAX_FILE_LEN)

Maximum file path (not including file name)

Limits:

Definition at line 43 of file default_cf_mission_cfg.h.

12.6 apps/cf/config/default_cf_msg.h File Reference

```
#include "cf_interface_cfg.h"
#include "cf_fcnodes.h"
#include "cf_msgdefs.h"
#include "cf_msgstruct.h"
```

Macros

- #define CF_COMPOUND_KEY (254)
- #define CF_ALL_CHANNELS (255)
- #define CF_ALL_POLLDIRS (CF_ALL_CHANNELS)

12.6.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message data types.

This is a compatibility header for the "cf_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.6.2 Macro Definition Documentation

12.6.2.1 CF_ALL_CHANNELS #define CF_ALL_CHANNELS (255)

Definition at line 41 of file default_cf_msg.h.

12.6.2.2 CF_ALL_POLLDIRS #define CF_ALL_POLLDIRS (CF_ALL_CHANNELS)

Definition at line 42 of file default_cf_msg.h.

12.6.2.3 CF_COMPOUND_KEY #define CF_COMPOUND_KEY (254)

Definition at line 40 of file default_cf_msg.h.

12.7 apps/cf/config/default_cf_msgdefs.h File Reference

```
#include "common_types.h"
#include "cf_extern_typedefs.h"
```

Data Structures

- struct [CF_HkCmdCounters](#)
Housekeeping command counters.
- struct [CF_HkSent](#)
Housekeeping sent counters.
- struct [CF_HkRecv](#)
Housekeeping received counters.
- struct [CF_HkFault](#)
Housekeeping fault counters.
- struct [CF_HkCounters](#)
Housekeeping counters.
- struct [CF_HkChannel_Data](#)
Housekeeping channel data.
- struct [CF_HkPacket_Payload](#)
Housekeeping packet.
- struct [CF_EotPacket_Payload](#)
End of transaction packet.
- union [CF_UnionArgs_Payload](#)
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.
- struct [CF_GetParam_Payload](#)
Get parameter command structure.
- struct [CF_SetParam_Payload](#)
Set parameter command structure.
- struct [CF_TxFile_Payload](#)
Transmit file command structure.
- struct [CF_WriteQueue_Payload](#)
Write Queue command structure.
- struct [CF_Transaction_Payload](#)
Transaction command structure.

TypeDefs

- **typedef struct CF_HkCmdCounters CF_HkCmdCounters_t**
Housekeeping command counters.
- **typedef struct CF_HkSent CF_HkSent_t**
Housekeeping sent counters.
- **typedef struct CF_HkRecv CF_HkRecv_t**
Housekeeping received counters.
- **typedef struct CF_HkFault CF_HkFault_t**
Housekeeping fault counters.
- **typedef struct CF_HkCounters CF_HkCounters_t**
Housekeeping counters.
- **typedef struct CF_HkChannel_Data CF_HkChannel_Data_t**
Housekeeping channel data.
- **typedef struct CF_HkPacket_Payload CF_HkPacket_Payload_t**
Housekeeping packet.
- **typedef struct CF_EotPacket_Payload CF_EotPacket_Payload_t**
End of transaction packet.
- **typedef union CF_UnionArgs_Payload CF_UnionArgs_Payload_t**
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.
- **typedef struct CF_GetParam_Payload CF_GetParam_Payload_t**
Get parameter command structure.
- **typedef struct CF_SetParam_Payload CF_SetParam_Payload_t**
Set parameter command structure.
- **typedef struct CF_TxFile_Payload CF_TxFile_Payload_t**
Transmit file command structure.
- **typedef struct CF_WriteQueue_Payload CF_WriteQueue_Payload_t**
Write Queue command structure.
- **typedef struct CF_Transaction_Payload CF_Transaction_Payload_t**
Transaction command structure.

Enumerations

- **enum CF_Reset_t {**
CF_Reset_all = 0, CF_Reset_command = 1, CF_Reset_fault = 2, CF_Reset_up = 3,
CF_Reset_down = 4 }
IDs for use for Reset cmd.
- **enum CF_Type_t { CF_Type_all = 0, CF_Type_up = 1, CF_Type_down = 2 }**
Type IDs for use for Write Queue cmd.
- **enum CF_Queue_t { CF_Queue_pend = 0, CF_Queue_active = 1, CF_Queue_history = 2, CF_Queue_all = 3 }**
Queue IDs for use for Write Queue cmd.
- **enum CF_GetSet_ValueID_t {**
CF_GetSet_ValueID_ticks_per_second, CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup, CF_GetSet_ValueID_ack_timer_s,
CF_GetSet_ValueID_nak_timer_s,
CF_GetSet_ValueID_inactivity_timer_s, CF_GetSet_ValueID_outgoing_file_chunk_size, CF_GetSet_ValueID_ack_limit,
CF_GetSet_ValueID_nak_limit,
CF_GetSet_ValueID_local_eid, CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup, CF_GetSet_ValueID_MAX
}
Parameter IDs for use with Get/Set parameter messages.

12.7.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message payload content definitions.

12.8 apps/cf/config/default_cf_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cf_topicids.h"
```

Macros

- #define CF_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CMD_TOPICID)
Message ID for commands.
- #define CF_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_SEND_HK_TOPICID)
Message ID to request housekeeping telemetry.
- #define CF_WAKE_UP_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_WAKE_UP_TOPICID)
Message ID for waking up the processing cycle.
- #define CF_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_HK_TLM_TOPICID)
Message ID for housekeeping telemetry.
- #define CF_EOT_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_EOT_TLM_TOPICID)
Message ID for end of transaction telemetry.
- #define CF_CH0_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_TX_TOPICID)
- #define CF_CH1_TX_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_TX_TOPICID)
- #define CF_CH0_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH0_RX_TOPICID)
- #define CF_CH1_RX_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_CF_CH1_RX_TOPICID)

12.8.1 Detailed Description

CFS CFDP (CF) Application Message IDs

12.9 apps/cf/config/default_cf_msgstruct.h File Reference

```
#include "cf_msgdefs.h"
#include "cf_mission_cfg.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- struct **CF_HkPacket**
Housekeeping packet.
- struct **CF_EotPacket**
End of transaction packet.
- struct **CF_NoopCmd**
Noop command structure.
- struct **CF_EnableEngineCmd**
EnableEngine command structure.
- struct **CF_DisableEngineCmd**
DisableEngine command structure.
- struct **CF_ResetCountersCmd**

- struct **CF_FreezeCmd**
Freeze command structure.
- struct **CF_ThawCmd**
Thaw command structure.
- struct **CF_EnableDequeueCmd**
EnableDequeue command structure.
- struct **CF_DisableDequeueCmd**
DisableDequeue command structure.
- struct **CF_EnableDirPollingCmd**
EnableDirPolling command structure.
- struct **CF_DisableDirPollingCmd**
DisableDirPolling command structure.
- struct **CF_PurgeQueueCmd**
PurgeQueue command structure.
- struct **CF_GetParamCmd**
Get parameter command structure.
- struct **CF_SetParamCmd**
Set parameter command structure.
- struct **CF_TxFileCmd**
Transmit file command structure.
- struct **CF_WriteQueueCmd**
Write Queue command structure.
- struct **CF_PlaybackDirCmd**
Playback directory command structure.
- struct **CF_SuspendCmd**
Suspend command structure.
- struct **CF_ResumeCmd**
Resume command structure.
- struct **CF_CancelCmd**
Cancel command structure.
- struct **CF_AbandonCmd**
Abandon command structure.
- struct **CF_SendHkCmd**
Send Housekeeping Command.
- struct **CF_WakeupCmd**
Wake Up Command.

Typedefs

- typedef struct **CF_HkPacket** **CF_HkPacket_t**
Housekeeping packet.
- typedef struct **CF_EotPacket** **CF_EotPacket_t**
End of transaction packet.
- typedef struct **CF_NoopCmd** **CF_NoopCmd_t**
Noop command structure.
- typedef struct **CF_EnableEngineCmd** **CF_EnableEngineCmd_t**
EnableEngine command structure.

- `typedef struct CF_DisableEngineCmd CF_DisableEngineCmd_t`
DisableEngine command structure.
- `typedef struct CF_ResetCountersCmd CF_ResetCountersCmd_t`
Reset command structure.
- `typedef struct CF_FreezeCmd CF_FreezeCmd_t`
Freeze command structure.
- `typedef struct CF_ThawCmd CF_ThawCmd_t`
Thaw command structure.
- `typedef struct CF_EnableDequeueCmd CF_EnableDequeueCmd_t`
EnableDequeue command structure.
- `typedef struct CF_DisableDequeueCmd CF_DisableDequeueCmd_t`
DisableDequeue command structure.
- `typedef struct CF_EnableDirPollingCmd CF_EnableDirPollingCmd_t`
EnableDirPolling command structure.
- `typedef struct CF_DisableDirPollingCmd CF_DisableDirPollingCmd_t`
DisableDirPolling command structure.
- `typedef struct CF_PurgeQueueCmd CF_PurgeQueueCmd_t`
PurgeQueue command structure.
- `typedef struct CF_GetParamCmd CF_GetParamCmd_t`
Get parameter command structure.
- `typedef struct CF_SetParamCmd CF_SetParamCmd_t`
Set parameter command structure.
- `typedef struct CF_TxFileCmd CF_TxFileCmd_t`
Transmit file command structure.
- `typedef struct CF_WriteQueueCmd CF_WriteQueueCmd_t`
Write Queue command structure.
- `typedef struct CF_PlaybackDirCmd CF_PlaybackDirCmd_t`
Playback directory command structure.
- `typedef struct CF_SuspendCmd CF_SuspendCmd_t`
Suspend command structure.
- `typedef struct CF_ResumeCmd CF_ResumeCmd_t`
Resume command structure.
- `typedef struct CF_CancelCmd CF_CancelCmd_t`
Cancel command structure.
- `typedef struct CF_AbandonCmd CF_AbandonCmd_t`
Abandon command structure.
- `typedef struct CF_SendHkCmd CF_SendHkCmd_t`
Send Housekeeping Command.
- `typedef struct CF_WakeupCmd CF_WakeupCmd_t`
Wake Up Command.

12.9.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message data types.

Note

Constants and enumerated types related to these message structures are defined in cf_msgdefs.h.

12.10 apps/cf/config/default_cf_platform_cfg.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_internal_cfg.h"
```

Macros

- `#define CF_TOTAL_CHUNKS (CF_NAK_MAX_SEGMENTS * 4)`
Total number of chunks (tx, rx, all channels)
- `#define CF_MISSION_REV 0`
Mission specific version number.

12.10.1 Detailed Description

CFS CFDP (CF) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.10.2 Macro Definition Documentation

12.10.2.1 CF_MISSION_REV `#define CF_MISSION_REV 0`

Mission specific version number.

Description:

An application 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 such that missions can manage as a configuration definition

Limits:

Must be defined as a numeric value that is greater than or equal to zero.

Definition at line 70 of file default_cf_platform_cfg.h.

12.10.2.2 CF_TOTAL_CHUNKS `#define CF_TOTAL_CHUNKS (CF_NAK_MAX_SEGMENTS * 4)`

Total number of chunks (tx, rx, all channels)

Description:

Must be equal to the sum of all values input in CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION and CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION.

Limits:

Definition at line 54 of file default_cf_platform_cfg.h.

12.11 apps/cf/config/default_cf_tbl.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_tbldefs.h"
#include "cf_tblstruct.h"
```

12.11.1 Detailed Description

Specification for the CFS CFDP (CF) table structures

Note

Constants and enumerated types related to these table structures are defined in cf_tbldefs.h.

12.12 apps/cf/config/default_cf_tbldefs.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_extern_typedefs.h"
#include "cf_tblstruct.h"
```

Data Structures

- struct [CF_PollDir](#)
Configuration entry for directory polling.
- struct [CF_ChannelConfig](#)
Configuration entry for CFDP channel.

TypeDefs

- typedef struct [CF_PollDir](#) [CF_PollDir_t](#)
Configuration entry for directory polling.
- typedef struct [CF_ChannelConfig](#) [CF_ChannelConfig_t](#)
Configuration entry for CFDP channel.

12.12.1 Detailed Description

Specification for the CFS CFDP (CF) table related constant definitions.

Note

These Macro definitions have been put in this file (instead of cf_tbl.h). DO NOT PUT ANY TYPEDEFS OR STRUCTURE DEFINITIONS IN THIS FILE! ADD THEM TO cf_tbl.h IF NEEDED!

12.12.2 Typedef Documentation

12.12.2.1 [CF_ChannelConfig_t](#) [typedef struct CF_ChannelConfig CF_ChannelConfig_t](#)

Configuration entry for CFDP channel.

12.12.2.2 [CF_PollDir_t](#) [typedef struct CF_PollDir CF_PollDir_t](#)

Configuration entry for directory polling.

12.13 apps/cf/config/default_cf_tblstruct.h File Reference

```
#include "common_types.h"
#include "cf_mission_cfg.h"
#include "cf_tbldefs.h"
```

Data Structures

- struct [CF_ConfigTable](#)

Top-level CFDP configuration structure.

TypeDefs

- typedef struct [CF_ConfigTable](#) [CF_ConfigTable_t](#)

Top-level CFDP configuration structure.

12.13.1 Detailed Description

Specification for the CFS CFDP (CF) table structures
Provides default definitions for HK table structures

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.13.2 Typedef Documentation

12.13.2.1 [CF_ConfigTable_t](#) [typedef struct CF_ConfigTable CF_ConfigTable_t](#)

Top-level CFDP configuration structure.

12.14 apps/cf/config/default_cf_topicids.h File Reference

Macros

- #define [CFE_MISSION_CF_CMD_TOPICID](#) 0xB3
Message ID for commands.
- #define [CFE_MISSION_CF_SEND_HK_TOPICID](#) 0xB4
Message ID to request housekeeping telemetry.
- #define [CFE_MISSION_CF_WAKE_UP_TOPICID](#) 0xB5
Message ID for waking up the processing cycle.
- #define [CFE_MISSION_CF_HK_TLM_TOPICID](#) 0xB0
Message ID for housekeeping telemetry.
- #define [CFE_MISSION_CF_EOT_TLM_TOPICID](#) 0xB3
Message ID for end of transaction telemetry.
- #define [CFE_MISSION_CF_CH0_TX_TOPICID](#) 0xB4
- #define [CFE_MISSION_CF_CH1_TX_TOPICID](#) 0xB5
- #define [CFE_MISSION_CF_CH0_RX_TOPICID](#) 0xB6
- #define [CFE_MISSION_CF_CH1_RX_TOPICID](#) 0xB7

12.14.1 Detailed Description

CFDP (CF) Application Topic IDs

12.14.2 Macro Definition Documentation

12.14.2.1 CFE_MISSION_CF_CH0_RX_TOPICID `#define CFE_MISSION_CF_CH0_RX_TOPICID 0xB6`
Definition at line 50 of file default_cf_topicids.h.

12.14.2.2 CFE_MISSION_CF_CH0_TX_TOPICID `#define CFE_MISSION_CF_CH0_TX_TOPICID 0xB4`
Definition at line 48 of file default_cf_topicids.h.

12.14.2.3 CFE_MISSION_CF_CH1_RX_TOPICID `#define CFE_MISSION_CF_CH1_RX_TOPICID 0xB7`
Definition at line 51 of file default_cf_topicids.h.

12.14.2.4 CFE_MISSION_CF_CH1_TX_TOPICID `#define CFE_MISSION_CF_CH1_TX_TOPICID 0xB5`
Definition at line 49 of file default_cf_topicids.h.

12.14.2.5 CFE_MISSION_CF_CMD_TOPICID `#define CFE_MISSION_CF_CMD_TOPICID 0xB3`
Message ID for commands.
Definition at line 33 of file default_cf_topicids.h.

12.14.2.6 CFE_MISSION_CF_EOT_TLM_TOPICID `#define CFE_MISSION_CF_EOT_TLM_TOPICID 0xB3`
Message ID for end of transaction telemetry.
Definition at line 37 of file default_cf_topicids.h.

12.14.2.7 CFE_MISSION_CF_HK_TLM_TOPICID `#define CFE_MISSION_CF_HK_TLM_TOPICID 0xB0`
Message ID for housekeeping telemetry.
Definition at line 36 of file default_cf_topicids.h.

12.14.2.8 CFE_MISSION_CF_SEND_HK_TOPICID `#define CFE_MISSION_CF_SEND_HK_TOPICID 0xB4`
Message ID to request housekeeping telemetry.
Definition at line 34 of file default_cf_topicids.h.

12.14.2.9 CFE_MISSION_CF_WAKE_UP_TOPICID `#define CFE_MISSION_CF_WAKE_UP_TOPICID 0xB5`
Message ID for waking up the processing cycle.
Definition at line 35 of file default_cf_topicids.h.

12.15 apps/cf/docs/dox_src/cfs_cf.dox File Reference

12.16 apps/cf/fsw/inc(cf_events.h File Reference

Macros

- #define CF_INIT_INF_EID (20)
CF Initialization Event ID.
- #define CF_INIT_TBL_CHECK_REL_ERR_EID (21)
CF Check Table Release Address Failed Event ID.
- #define CF_INIT_TBL_CHECK_MAN_ERR_EID (22)
CF Check Table Manage Failed Event ID.
- #define CF_INIT_TBL_CHECK_GA_ERR_EID (23)
CF Check Table Get Address Failed Event ID.
- #define CF_INIT_TBL_REG_ERR_EID (24)
CF Table Registration At Initialization Failed Event ID.
- #define CF_INIT_TBL_LOAD_ERR_EID (25)
CF Table Load At Initialization Failed Event ID.
- #define CF_INIT_TBL_MANAGE_ERR_EID (26)
CF Table Manage At Initialization Failed Event ID.
- #define CF_INIT_TBL_GETADDR_ERR_EID (27)
CF Table Get Address At Initialization Failed Event ID.
- #define CF_MID_ERR_EID (28)
CF Message ID Invalid Event ID.
- #define CF_INIT_MSG_RECV_ERR_EID (29)
CF SB Receive Buffer Failed Event ID.
- #define CF_INIT_SEM_ERR_EID (30)
CF Channel Semaphore Initialization Failed Event ID.
- #define CF_CR_CHANNEL_PIPE_ERR_EID (31)
CF Channel Create Pipe Failed Event ID.
- #define CF_INIT_SUB_ERR_EID (32)
CF Channel Message Subscription Failed Event ID.
- #define CF_INIT_TPS_ERR_EID (33)
CF Ticks Per Second Config Table Validation Failed Event ID.
- #define CF_INIT_CRC_ALIGN_ERR_EID (34)
CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.
- #define CF_INIT_OUTGOING_SIZE_ERR_EID (35)
CF Outgoing Chunk Size Config Table Validation Failed Event ID.
- #define CF_CR_PIPE_ERR_EID (36)
CF Create SB Command Pipe at Initialization Failed Event ID.
- #define CF_PDU_MD_RECVD_INF_EID (40)
CF Metadata PDU Received Event ID.
- #define CF_PDU_SHORT_HEADER_ERR_EID (41)
CF PDU Header Too Short Event ID.
- #define CF_PDU_MD_SHORT_ERR_EID (43)
CF Metadata PDU Too Short Event ID.
- #define CF_PDU_INVALID_SRC_LEN_ERR_EID (44)
CF Metadata PDU Source Filename Length Invalid Event ID.
- #define CF_PDU_INVALID_DST_LEN_ERR_EID (45)

- #define CF_PDU_FD_SHORT_ERR_EID (46)
CF File Data PDU Too Short Event ID.
- #define CF_PDU_EOF_SHORT_ERR_EID (47)
CF End-Of-File PDU Too Short Event ID.
- #define CF_PDU_ACK_SHORT_ERR_EID (48)
CF Acknowledgment PDU Too Short Event ID.
- #define CF_PDU_FIN_SHORT_ERR_EID (49)
CF Finished PDU Too Short Event ID.
- #define CF_PDU_NAK_SHORT_ERR_EID (50)
CF Negative Acknowledgment PDU Too Short Event ID.
- #define CF_PDU_FD_UNSUPPORTED_ERR_EID (54)
CF File Data PDU Unsupported Option Event ID.
- #define CF_PDU_LARGE_FILE_ERR_EID (55)
CF PDU Header Large File Flag Set Event ID.
- #define CF_PDU_TRUNCATION_ERR_EID (56)
CF PDU Header Field Truncation.
- #define CF_RESET_FREED_XACT_DBG_EID (59)
Attempt to reset a transaction that has already been freed.
- #define CF_CFDP_RX_DROPPED_ERR_EID (60)
CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.
- #define CF_CFDP_INVALID_DST_ERR_EID (61)
CF PDU Received With Invalid Destination Entity ID Event ID.
- #define CF_CFDP_IDLE_MD_ERR_EID (62)
CF Invalid Metadata PDU Received On Idle Transaction Event ID.
- #define CF_CFDP_FD_UNHANDLED_ERR_EID (63)
CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.
- #define CF_CFDP_MAX_CMD_TX_ERR_EID (64)
CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.
- #define CF_CFDP_OPENDIR_ERR_EID (65)
CF Playback/Polling Directory Open Failed Event ID.
- #define CF_CFDP_DIR_SLOT_ERR_EID (66)
CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.
- #define CF_CFDP_NO_MSG_ERR_EID (67)
CF No Message Buffer Available Event ID.
- #define CF_CFDP_CLOSE_ERR_EID (68)
CF Close File Failed Event ID.
- #define CF_CFDP_R_REQUEST_MD_INF_EID (70)
CF Requesting RX Metadata Event ID.
- #define CF_CFDP_R_TEMP_FILE_INF_EID (71)
CF Creating Temp File For RX Transaction Without Metadata PDU.
- #define CF_CFDP_R_NAK_LIMIT_ERR_EID (72)
CF RX Transaction NAK Limit Reached Event ID.
- #define CF_CFDP_R_ACK_LIMIT_ERR_EID (73)
CF RX Transaction ACK Limit Reached Event ID.
- #define CF_CFDP_R_CRC_ERR_EID (74)
CF RX Transaction CRC Mismatch Event ID.

- #define CF_CFDP_R_SEEK_FD_ERR_EID (75)
CF RX File Data PDU Seek Failed Event ID.
- #define CF_CFDP_R_SEEK_CRC_ERR_EID (76)
CF RX Class 2 CRC Seek Failed Event ID.
- #define CF_CFDP_R_WRITE_ERR_EID (77)
CF RX File Data PDU Write Failed Event ID.
- #define CF_CFDP_R_SIZE_MISMATCH_ERR_EID (78)
CF RX End-Of-File PDU File Size Mismatch Event ID.
- #define CF_CFDP_R_PDU_EOF_ERR_EID (79)
CF Invalid End-Of-File PDU Event ID.
- #define CF_CFDP_R_CREAT_ERR_EID (80)
CF RX Transaction File Create Failed Event ID.
- #define CF_CFDP_R_PDU_FINACK_ERR_EID (81)
CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.
- #define CF_CFDP_R_EOF_MD_SIZE_ERR_EID (82)
CF RX Class 2 Metadata PDU Size Mismatch Event ID.
- #define CF_CFDP_R_RENAME_ERR_EID (83)
CF RX Class 2 Metadata PDU File Rename Failed Event ID.
- #define CF_CFDP_R_OPEN_ERR_EID (84)
CF RX Class 2 Metadata PDU File Open Failed Event ID.
- #define CF_CFDP_R_PDU_MD_ERR_EID (85)
CF Invalid Out-of-order Metadata PDU Received Event ID.
- #define CF_CFDP_R_READ_ERR_EID (86)
CF Class 2 CRC Read From File Failed Event ID.
- #define CF_CFDP_R_DC_INV_ERR_EID (87)
CF RX Invalid File Directive PDU Code Received Event ID.
- #define CF_CFDP_R_INACT_TIMER_ERR_EID (88)
CF RX Inactivity Timer Expired Event ID.
- #define CF_CFDP_S_START_SEND_INF_EID (90)
CF TX Initiated Event ID.
- #define CF_CFDP_S_SEEK_FD_ERR_EID (91)
CF TX File Data PDU Seek Failed Event ID.
- #define CF_CFDP_S_READ_ERR_EID (92)
CF TX File Data PDU Read Failed Event ID.
- #define CF_CFDP_S_SEND_FD_ERR_EID (93)
CF TX File Data PDU Send Failed Event ID.
- #define CF_CFDP_S_ALREADY_OPEN_ERR_EID (94)
CF TX Metadata PDU File Already Open Event ID.
- #define CF_CFDP_S_OPEN_ERR_EID (95)
CF TX Metadata PDU File Open Failed Event ID.
- #define CF_CFDP_S_SEEK_END_ERR_EID (96)
CF TX Metadata PDU File Seek End Failed Event ID.
- #define CF_CFDP_S_SEEK_BEG_ERR_EID (97)
CF TX Metadata PDU File Seek Beginning Failed Event ID.
- #define CF_CFDP_S_SEND_MD_ERR_EID (98)
CF TX Metadata PDU Send Failed Event ID.
- #define CF_CFDP_S_INVALID_SR_ERR_EID (100)

- #define CF_CFDP_S_PDU_NAK_ERR_EID (101)
CF TX Received NAK PDU Bad Segment Request Event ID.
- #define CF_CFDP_S_PDU_EOF_ERR_EID (102)
CF TX Received EOF ACK PDU Invalid Event ID.
- #define CF_CFDP_S_EARLY_FIN_ERR_EID (103)
CF TX Received Early FIN PDU Event ID.
- #define CF_CFDP_S_DC_INV_ERR_EID (104)
CF Invalid TX File Directive PDU Code Event ID.
- #define CF_CFDP_S_NON_FD_PDU_ERR_EID (105)
CF Received TX Non-File Directive PDU Event ID.
- #define CF_CFDP_S_ACK_LIMIT_ERR_EID (106)
CF TX EOF PDU Send Limit Reached Event ID.
- #define CF_CFDP_S_INACT_TIMER_ERR_EID (107)
CF TX Inactivity Timer Expired Event ID.
- #define CF_NOOP_INF_EID (110)
CF NOOP Command Received Event ID.
- #define CF_RESET_INF_EID (111)
CF Reset Counters Command Received Event ID.
- #define CF_CMD_GETSET1_INF_EID (112)
CF Set Parameter Command Received Event ID.
- #define CF_CMD_GETSET2_INF_EID (113)
CF Get Parameter Command Received Event ID.
- #define CF_CMD_SUSPRES_INF_EID (114)
CF Suspend/Resume Command Received Event ID.
- #define CF_CMD_WQ_INF_EID (115)
CF Write Queue Command Received Event ID.
- #define CF_CMD_ENABLE_ENGINE_INF_EID (116)
CF Enable Engine Command Received Event ID.
- #define CF_CMD_DISABLE_ENGINE_INF_EID (117)
CF Disable Engine Command Received Event ID.
- #define CF_CMD_TX_FILE_INF_EID (118)
CF Transfer File Command Received Event ID.
- #define CF_CMD_PLAYBACK_DIR_INF_EID (119)
CF Playback Directory Command Received Event ID.
- #define CF_CMD_FREEZE_INF_EID (120)
CF Freeze Command Received Event ID.
- #define CF_CMD_THAW_INF_EID (121)
CF Thaw Command Received Event ID.
- #define CF_CMD_CANCEL_INF_EID (122)
CF Cancel Command Received Event ID.
- #define CF_CMD_ABANDON_INF_EID (123)
CF Abandon Command Received Event ID.
- #define CF_CMD_ENABLE_DEQUEUE_INF_EID (124)
CF Enable Dequeue Command Received Event ID.
- #define CF_CMD_DISABLE_DEQUEUE_INF_EID (125)
CF Disable Dequeue Command Received Event ID.

- #define CF_CMD_ENABLE_POLLDIR_INF_EID (126)
CF Enable Polldir Command Received Event ID.
- #define CF_CMD_DISABLE_POLLDIR_INF_EID (127)
CF Disable Polldir Command Received Event ID.
- #define CF_CMD_PURGE_QUEUE_INF_EID (128)
CF Purge Queue Command Received Event ID.
- #define CF_CMD_RESET_INVALID_ERR_EID (129)
CF Reset Counters Command Invalid Event ID.
- #define CF_CMD_CHAN_PARAM_ERR_EID (130)
CF Command Channel Invalid Event ID.
- #define CF_CMD_TRANS_NOT_FOUND_ERR_EID (131)
CF Command Transaction Invalid Event ID.
- #define CF_CMD_TSN_CHAN_INVALID_ERR_EID (132)
CF Command All Transaction Channel Invalid Event ID.
- #define CF_CMD_SUSPRES_SAME_INF_EID (133)
CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.
- #define CF_CMD_SUSPRES_CHAN_ERR_EID (134)
CF Suspend/Resume Command No Matching Transaction Event ID.
- #define CF_CMD_POLLDIR_INVALID_ERR_EID (135)
CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.
- #define CF_CMD_PURGE_ARG_ERR_EID (136)
CF Purge Queue Command Invalid Argument Event ID.
- #define CF_CMD_WQ_CHAN_ERR_EID (137)
CF Write Queue Command Invalid Channel Event ID.
- #define CF_CMD_WQ_ARGS_ERR_EID (138)
CF Write Queue Command Invalid Queue Event ID.
- #define CF_CMD_WQ_OPEN_ERR_EID (139)
CF Write Queue Command File Open Failed Event ID.
- #define CF_CMD_WQ_WRITEQ_RX_ERR_EID (140)
CF Write Queue Command RX Active File Write Failed Event ID.
- #define CF_CMD_WQ_WRITEHIST_RX_ERR_EID (141)
CF Write Queue Command RX History File Write Failed Event ID.
- #define CF_CMD_WQ_WRITEQ_TX_ERR_EID (142)
CF Write Queue Command TX Active File Write Failed Event ID.
- #define CF_CMD_WQ_WRITEQ_PEND_ERR_EID (143)
CF Write Queue Command TX Pending File Write Failed Event ID.
- #define CF_CMD_WQ_WRITEHIST_TX_ERR_EID (144)
CF Write Queue Command TX History File Write Failed Event ID.
- #define CF_CMD_GETSET_VALIDATE_ERR_EID (145)
CF Set Parameter Command Parameter Validation Failed Event ID.
- #define CF_CMD_GETSET_PARAM_ERR_EID (146)
CF Set/Get Parameter Command Invalid Parameter ID Event ID.
- #define CF_CMD_GETSET_CHAN_ERR_EID (147)
CF Set/Get Parameter Command Invalid Channel Event ID.
- #define CF_CMD_ENABLE_ENGINE_ERR_EID (148)
CF Enable Engine Command Failed Event ID.
- #define CF_CMD_ENG_ALREADY_ENA_INF_EID (149)

- #define CF_CMD_ENG_ALREADY_DIS_INF_EID (150)
CF Enable Engine Command Engine Already Enabled Event ID.
- #define CF_CMD_LEN_ERR_EID (151)
CF Command Length Verification Failed Event ID.
- #define CF_CC_ERR_EID (152)
CF Command Code Invalid Event ID.
- #define CF_CMD_WHIST_WRITE_ERR_EID (153)
CF Write Entry To File Failed Event ID.
- #define CF_CMD_BAD_PARAM_ERR_EID (154)
CF Playback Dir Or TX File Command Bad Parameter Event ID.
- #define CF_CMD_CANCEL_CHAN_ERR_EID (155)
CF Cancel Command No Matching Transaction Event ID.
- #define CF_CMD_ABANDON_CHAN_ERR_EID (156)
CF Abandon Command No Matching Transaction Event ID.
- #define CF_CMD_TX_FILE_ERR_EID (157)
CF Transfer File Command Failed Event ID.
- #define CF_CMD_PLAYBACK_DIR_ERR_EID (158)
CF Playback Directory Command Failed Event ID.
- #define CF_CMD_FREEZE_ERR_EID (159)
CF Freeze Command Failed Event ID.
- #define CF_CMD_THAW_ERR_EID (160)
CF Thaw Command Failed Event ID.
- #define CF_CMD_ENABLE_DEQUEUE_ERR_EID (161)
CF Enable Dequeue Command Failed Event ID.
- #define CF_CMD_DISABLE_DEQUEUE_ERR_EID (162)
CF Disable Dequeue Command Failed Event ID.
- #define CF_CMD_ENABLE_POLLDIR_ERR_EID (163)
CF Enable Polldir Command Failed Event ID.
- #define CF_CMD_DISABLE_POLLDIR_ERR_EID (164)
CF Disable Polldir Command Failed Event ID.
- #define CF_CMD_PURGE_QUEUE_ERR_EID (165)
CF Purge Queue Command Failed Event ID.

12.16.1 Detailed Description

The CF Application event id definition header file

12.17 apps/cf/fsw/inc/cf_perfids.h File Reference

Macros

- #define CF_PERF_ID_APPMAIN (11)
Main application performance ID.
- #define CF_PERF_ID_FSEEK (12)
File seek performance ID.
- #define CF_PERF_ID_FOPEN (13)
File open performance ID.
- #define CF_PERF_ID_FCLOSE (14)

- `#define CF_PERF_ID_FREAD (15)`
File close performance ID.
- `#define CF_PERF_ID_FWRITE (16)`
File read performance ID.
- `#define CF_PERF_ID_CYCLE_ENG (17)`
File write performance ID.
- `#define CF_PERF_ID_DIRREAD (18)`
Cycle engine performance ID.
- `#define CF_PERF_ID_CREAT (19)`
Directory read performance ID.
- `#define CF_PERF_ID_RENAME (20)`
Create performance ID.
- `#define CF_PERF_ID_PDURCVD(x) (30 + x)`
PDU Received performance ID.
- `#define CF_PERF_ID_PDUSENT(x) (40 + x)`
PDU Sent performance ID.

12.17.1 Detailed Description

Define CF Performance IDs

12.18 apps/cf/fsw/src(cf_app.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_version.h"
#include "cf_dispatch.h"
#include "cf_tbl.h"
#include <string.h>
```

Functions

- `void CF_CheckTables (void)`
Checks to see if a table update is pending, and perform it.
- `CFE_Status_t CF_ValidateConfigTable (void *tbl_ptr)`
Validation function for config table.
- `CFE_Status_t CF_TableInit (void)`
Load the table on application start.
- `CFE_Status_t CF_AppInit (void)`
CF app init function.
- `void CF_AppMain (void)`
CF app entry point.

Variables

- [CF_AppData_t CF_AppData](#)

Singleton instance of the application global data.

12.18.1 Detailed Description

The CF Application main application source file

This file contains the functions that initialize the application and link all logic and functionality to the CFS.

12.18.2 Function Documentation

12.18.2.1 CF_AppInit() [CFE_Status_t CF_AppInit](#) (

`void`

CF app init function.

Description

Initializes all aspects of the CF application. Messages, pipes, events, table, and the CFDP engine.

Assumptions, External Events, and Notes:

This must only be called once.

Return values

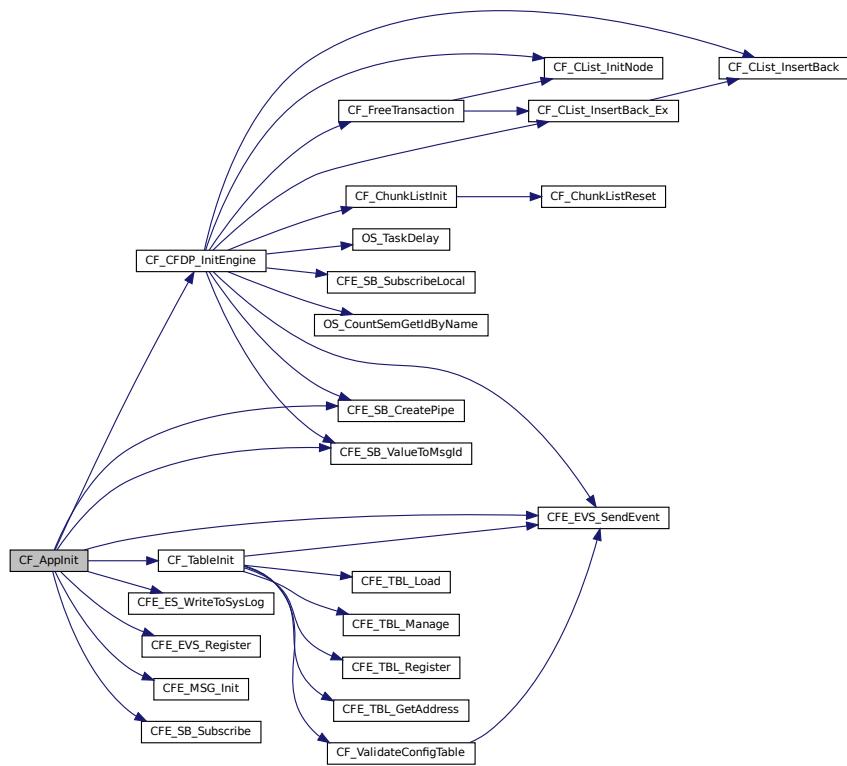
CFE_SUCCESS	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

Definition at line 189 of file cf_app.c.

References CF_AppData, CF_CFDP_InitEngine(), CF_CMD_MID, CF_CR_PIPE_ERR_EID, CF_HK_TLM_MID, CF_INIT_INF_EID, CF_MAJOR_VERSION, CF_MINOR_VERSION, CF_MISSION_REV, CF_PIPE_DEPTH, CF_PIPE_NAME, CF_REVISION, CF_SEND_HK_MID, CF_TableInit(), CF_WAKE_UP_MID, CFE_ES_RunStatus_APP_RUN, CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CF_AppData_t::CmdPipe, CF_AppData_t::hk, CF_AppData_t::RunStatus, and CF_HkPacket::TelemetryHeader.

Referenced by CF_AppMain().

Here is the call graph for this function:



12.18.2.2 CF_AppMain()

```
void CF_AppMain (
    void )
```

CF app entry point.

Description

Main entry point of CF application. Calls the init function and manages the app run loop.

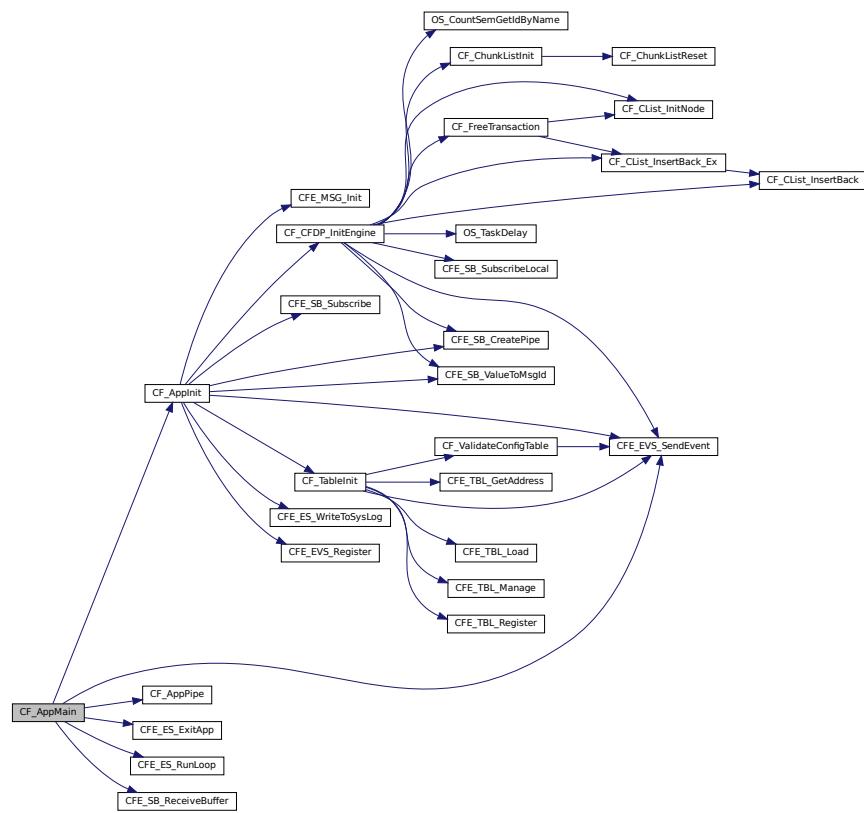
Assumptions, External Events, and Notes:

This must only be called once.

Definition at line 256 of file cf_app.c.

References CF_AppData, CF_AppInit(), CF_AppPipe(), CF_INIT_MSG_RECV_ERR_EID, CF_PERF_ID_APPMA_IN, CF_RCVMSG_TIMEOUT, CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_RunStatus_APP_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_NO_MESSAGE, CFE_SB_ReceiveBuffer(), CFE_SB_TIME_OUT, CFE_SUCCESS, CF_AppData_t::CmdPipe, and CF_AppData_t::RunStatus.

Here is the call graph for this function:



12.18.2.3 CF_CheckTables()

```
void CF_CheckTables(
    void )
```

Checks to see if a table update is pending, and perform it.

Description

Updates the table if the engine is disabled.

Assumptions, External Events, and Notes:

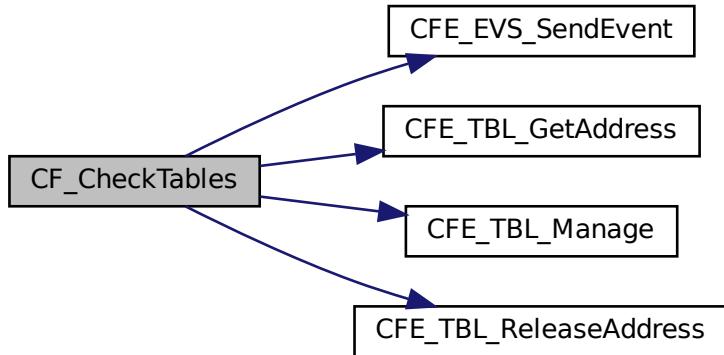
None

Definition at line 48 of file `cf_app.c`.

References `CF_AppData`, `CF_INIT_TBL_CHECK_GA_ERR_EID`, `CF_INIT_TBL_CHECK_MAN_ERR_EID`, `CF_INIT_TBL_CHECK_REL_ERR_EID`, `CFE_ES_RunStatus_APP_ERROR`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CFE_TBL_GetAddress()`, `CFE_TBL_Manage()`, `CFE_TBL_ReleaseAddress()`, `CF_AppData_t::config_handle`, `CF_AppData_t::config_table`, `CF_Engine::enabled`, `CF_AppData_t::engine`, and `CF_AppData_t::RunStatus`.

Referenced by `CF_SendHkCmd()`.

Here is the call graph for this function:



12.18.2.4 CF_TableInit() `CFE_Status_t` CF_TableInit (
 `void`)

Load the table on application start.

Assumptions, External Events, and Notes:

None

Return values

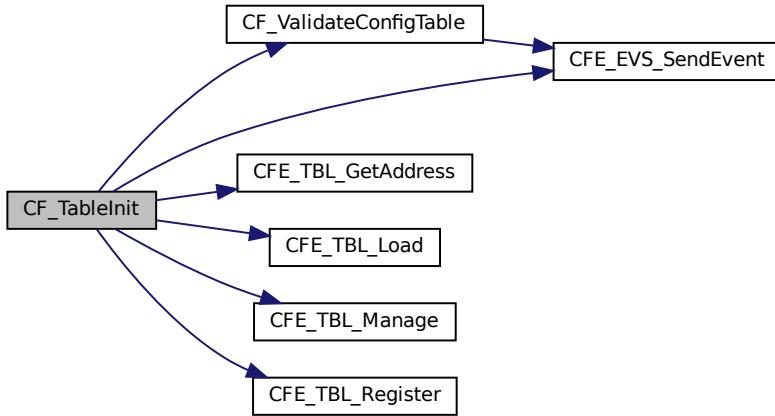
<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

Definition at line 134 of file cf_app.c.

References `CF_AppData`, `CF_CONFIG_TABLE_FILENAME`, `CF_CONFIG_TABLE_NAME`, `CF_INIT_TBL_GETADRERR_EID`, `CF_INIT_TBL_LOAD_ERR_EID`, `CF_INIT_TBL_MANAGE_ERR_EID`, `CF_INIT_TBL_REG_ERR_EID`, `CF_ValidateConfigTable()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CFE_TBL_GetAddress()`, `CFE_TBL_INFO_UPDATED`, `CFE_TBL_Load()`, `CFE_TBL_Manage()`, `CFE_TBL_OPT_LOAD_DUMP`, `CFE_TBL_OPT_SNGL_BUFFER`, `CFE_TBL_Register()`, `CFE_TBL_SRC_FILE`, `CF_AppData_t::config_handle`, and `CF_AppData_t::config_table`.

Referenced by `CF_AppInit()`.

Here is the call graph for this function:



12.18.2.5 CF_ValidateConfigTable() `CFE_Status_t CF_ValidateConfigTable (void *tbl_ptr)`

Validation function for config table.

Description

Checks that the config table being loaded has correct data.

Assumptions, External Events, and Notes:

None

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CFE_STATUS_VALIDATION_FAILURE</code>	if the config table fails one of the validation checks

Definition at line 101 of file cf_app.c.

References CF_INIT_CRC_ALIGN_ERR_EID, CF_INIT_OUTGOING_SIZE_ERR_EID, CF_INIT_TPS_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_STATUS_VALIDATION_FAILURE, CFE_SUCCESS, CF_ConfigTable::outgoing_file_chunk_size, CF_ConfigTable::rx_crc_calc_bytes_per_wakeup, and CF_ConfigTable::ticks_per_second.

Referenced by CF_TableInit().

Here is the call graph for this function:



12.18.3 Variable Documentation

12.18.3.1 CF_AppData CF_AppData_t CF_AppData

Singleton instance of the application global data.

Definition at line 40 of file cf_app.c.

Referenced by CF_AbandonCmd(), CF_AppInit(), CF_AppMain(), CF_AppPipe(), CF_CancelCmd(), CF_CFDP_AppendTlv(), CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), CF_CFDP_CycleEngine(), CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_HandleNotKeepFile(), CF_CFDP_InitEngine(), CF_CFDP_IsPollingDir(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir(), CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvDrop(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_ResetTransaction(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_Send(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), CF_CheckTables(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DisqueueTransaction(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DisableEngineCmd(), CF_DoEnableDisableDequeue(), CF_DoEnableDisablePolldir(), CF_DoFreezeThaw(), CF_DoPurgeQueue(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FindTransactionBySequenceNumberAllChannels(), CF_FreeTransaction(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_InsertSortPrio(), CF_MoveTransaction(), CF_NoopCmd(), CF_PlaybackDirCmd(), CF_ProcessGroundCommand(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_SendHkCmd(), CF_TableInit(), CF_ThawCmd(), CF_Timer_Sec2Ticks(), CF_TraverseAllTransactions_All_Channels(), CF_TsnChanAction(), CF_TxFileCmd(), CF_ValidateMaxOutgoingCmd(), and CF_WriteQueueCmd().

12.19 apps/cf/fsw/src(cf_app.h File Reference

```

#include "cfe.h"
#include "cf_msg.h"
#include "cf_tbl.h"
#include "cf_msgids.h"
#include "cf_tbldefs.h"
#include "cf_mission_cfg.h"
#include "cf_platform_cfg.h"
#include "cf_cfdp.h"
#include "cf_clist.h"

```

Data Structures

- struct [CF_AppData_t](#)
The CF application global state structure.

Macros

- #define [CF_PIPE_NAME](#) ("CF_CMD_PIPE")
The name of the application command pipe for CF.
- #define [CF_CHANNEL_PIPE_PREFIX](#) ("CF_CHAN_")
A common prefix for all data pipes for CF.

CF Error Codes

- #define [CF_ERROR](#) -1
Generic CF error return code.
- #define [CF_PDU_METADATA_ERROR](#) -2
Invalid metadata PDU.
- #define [CF_SHORT_PDU_ERROR](#) -3
PDU too short.
- #define [CF_REC_PDU_FSIZE_MISMATCH_ERROR](#) -4
Receive PDU: EOF file size mismatch.
- #define [CF_REC_PDU_BAD_EOF_ERROR](#) -5
Receive PDU: Invalid EOF packet.
- #define [CF_SEND_PDU_NO_BUF_AVAIL_ERROR](#) -6
Send PDU: No send buffer available, throttling limit reached.
- #define [CF_SEND_PDU_ERROR](#) -7
Send PDU: Send failed.

Functions

- void [CF_CheckTables](#) (void)
Checks to see if a table update is pending, and perform it.
- [CFE_Status_t](#) [CF_ValidateConfigTable](#) (void *tbl_ptr)
Validation function for config table.
- [CFE_Status_t](#) [CF_TableInit](#) (void)
Load the table on application start.
- [CFE_Status_t](#) [CF_AppInit](#) (void)
CF app init function.
- void [CF_AppMain](#) (void)
CF app entry point.

Variables

- [CF_AppData_t](#) [CF_AppData](#)
Singleton instance of the application global data.

12.19.1 Detailed Description

The CF Application main application header file

12.19.2 Macro Definition Documentation

12.19.2.1 CF_CHANNEL_PIPE_PREFIX `#define CF_CHANNEL_PIPE_PREFIX ("CF_CHAN_")`

A common prefix for all data pipes for CF.

Definition at line 67 of file cf_app.h.

12.19.2.2 CF_ERROR `#define CF_ERROR -1`

Generic CF error return code.

Definition at line 50 of file cf_app.h.

12.19.2.3 CF_PDU_METADATA_ERROR `#define CF_PDU_METADATA_ERROR -2`

Invalid metadata PDU.

Definition at line 51 of file cf_app.h.

12.19.2.4 CF_PIPE_NAME `#define CF_PIPE_NAME ("CF_CMD_PIPE")`

The name of the application command pipe for CF.

Definition at line 62 of file cf_app.h.

12.19.2.5 CF_REC_PDU_BAD_EOF_ERROR `#define CF_REC_PDU_BAD_EOF_ERROR -5`

Receive PDU: Invalid EOF packet.

Definition at line 54 of file cf_app.h.

12.19.2.6 CF_REC_PDU_FSIZE_MISMATCH_ERROR `#define CF_REC_PDU_FSIZE_MISMATCH_ERROR -4`

Receive PDU: EOF file size mismatch.

Definition at line 53 of file cf_app.h.

12.19.2.7 CF_SEND_PDU_ERROR `#define CF_SEND_PDU_ERROR -7`

Send PDU: Send failed.

Definition at line 56 of file cf_app.h.

12.19.2.8 CF_SEND_PDU_NO_BUF_AVAIL_ERROR `#define CF_SEND_PDU_NO_BUF_AVAIL_ERROR -6`

Send PDU: No send buffer available, throttling limit reached.

Definition at line 55 of file cf_app.h.

12.19.2.9 CF_SHORT_PDU_ERROR `#define CF_SHORT_PDU_ERROR -3`

PDU too short.

Definition at line 52 of file cf_app.h.

12.19.3 Function Documentation

12.19.3.1 CF_AppInit() `CFE_Status_t CF_AppInit (`
`void)`

CF app init function.

Description

Initializes all aspects of the CF application. Messages, pipes, events, table, and the CFDP engine.

Assumptions, External Events, and Notes:

This must only be called once.

Return values

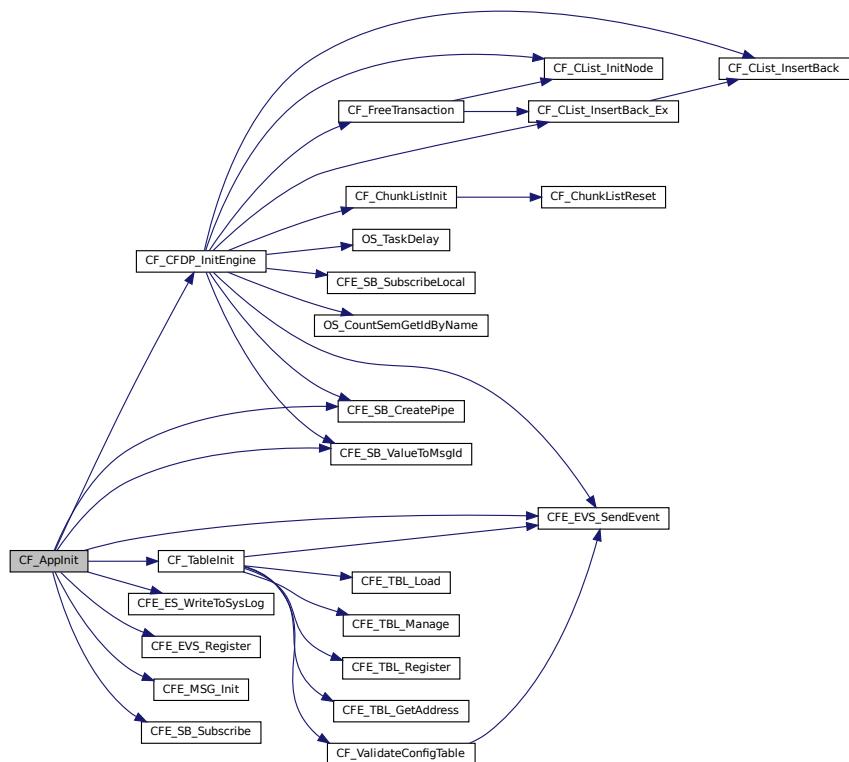
CFE_SUCCESS	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

Definition at line 189 of file cf_app.c.

References CF_AppData, CF_CFDP_InitEngine(), CF_CMD_MID, CF_CR_PIPE_ERR_EID, CF_HK_TLM_MID, CF_INIT_INF_EID, CF_MAJOR_VERSION, CF_MINOR_VERSION, CF_MISSION_REV, CF_PIPE_DEPTH, CF_PIPE_NAME, CF_REVISION, CF_SEND_HK_MID, CF_TableInit(), CF_WAKE_UP_MID, CFE_ES_RunStatus_APP_RUN, CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CF_AppData_t::CmdPipe, CF_AppData_t::hk, CF_AppData_t::RunStatus, and CF_HkPacket::TelemetryHeader.

Referenced by CF_AppMain().

Here is the call graph for this function:



```
12.19.3.2 CF_AppMain() void CF_AppMain (
    void )
```

CF app entry point.

Description

Main entry point of CF application. Calls the init function and manages the app run loop.

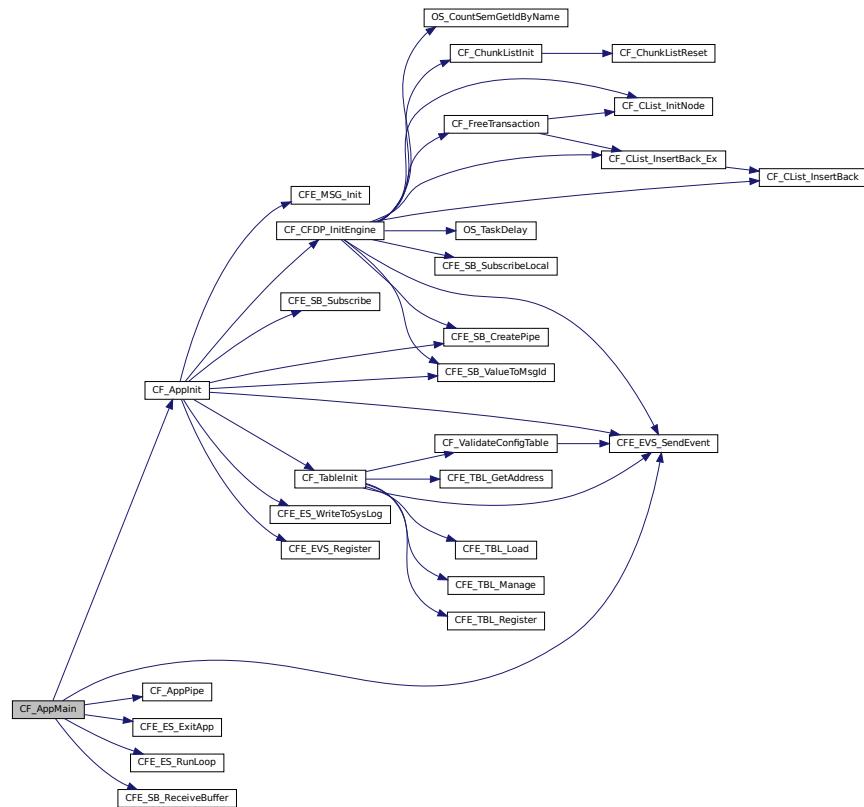
Assumptions, External Events, and Notes:

This must only be called once.

Definition at line 256 of file cf_app.c.

References CF_AppData, CF_AppInit(), CF_AppPipe(), CF_INIT_MSG_RECV_ERR_EID, CF_PERF_ID_APPMAIN, CF_RCVMSG_TIMEOUT, CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_RunStatus_APP_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_NO_MESSAGE, CFE_SB_ReceiveBuffer(), CFE_SB_TIME_OUT, CFE_SUCCESS, CF_AppData_t::CmdPipe, and CF_AppData_t::RunStatus.

Here is the call graph for this function:



```
12.19.3.3 CF_CheckTables() void CF_CheckTables (
    void )
```

Checks to see if a table update is pending, and perform it.

Description

Updates the table if the engine is disabled.

Assumptions, External Events, and Notes:

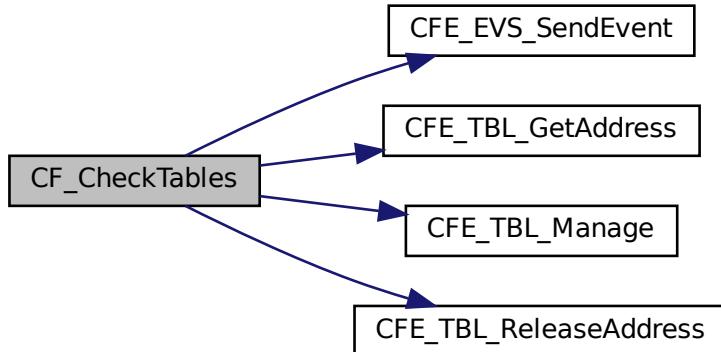
None

Definition at line 48 of file cf_app.c.

References CF_AppData, CF_INIT_TBL_CHECK_GA_ERR_EID, CF_INIT_TBL_CHECK_MAN_ERR_EID, CF_INIT_TBL_CHECK_REL_ERR_EID, CFE_ES_RunStatus_APP_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TBL_GetAddress(), CFE_TBL_Manage(), CFE_TBL_ReleaseAddress(), CF_AppData_t::config_handle, CF_AppData_t::config_table, CF_Engine::enabled, CF_AppData_t::engine, and CF_AppData_t::RunStatus.

Referenced by CF_SendHkCmd().

Here is the call graph for this function:



12.19.3.4 CF_TableInit() `CFE_Status_t CF_TableInit (void)`

Load the table on application start.

Assumptions, External Events, and Notes:

None

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

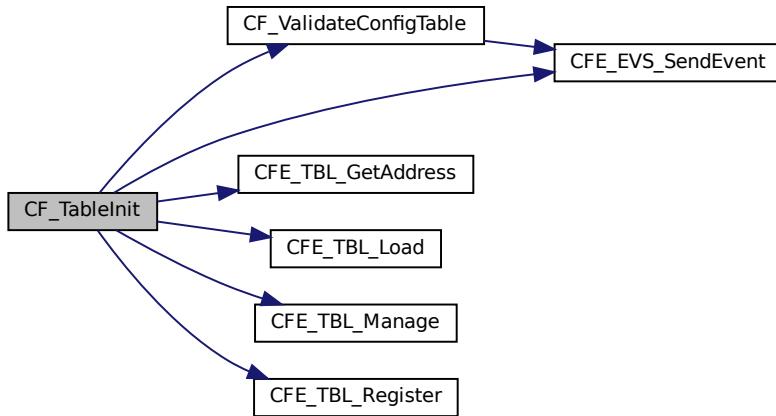
Definition at line 134 of file cf_app.c.

References CF_AppData, CF_CONFIG_TABLE_FILENAME, CF_CONFIG_TABLE_NAME, CF_INIT_TBL_GETAD

DR_ERR_EID, CF_INIT_TBL_LOAD_ERR_EID, CF_INIT_TBL_MANAGE_ERR_EID, CF_INIT_TBL_REG_ERR_EID, CF_ValidateConfigTable(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TBL_GetAddress(), CFE_TBL_INFO_UPDATED, CFE_TBL_Load(), CFE_TBL_Manage(), CFE_TBL_OPT_LOAD_DUMP, CFE_TBL_OPT_SNGL_BUFFER, CFE_TBL_Register(), CFE_TBL_SRC_FILE, CF_AppData_t::config_handle, and CF_AppData_t::config_table.

Referenced by CF_AppInit().

Here is the call graph for this function:



12.19.3.5 CF_ValidateConfigTable() `CFE_Status_t CF_ValidateConfigTable (void *tbl_ptr)`

Validation function for config table.

Description

Checks that the config table being loaded has correct data.

Assumptions, External Events, and Notes:

None

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CFE_STATUS_VALIDATION_FAILURE</code>	if the config table fails one of the validation checks

Definition at line 101 of file cf_app.c.

References CF_INIT_CRC_ALIGN_ERR_EID, CF_INIT_OUTGOING_SIZE_ERR_EID, CF_INIT_TPS_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_STATUS_VALIDATION_FAILURE, CFE_SUCCESS, CF_ConfigTable::outgoing_file_chunk_size, CF_ConfigTable::rx_crc_calc_bytes_per_wakeup, and CF_ConfigTable::ticks_per_second.

Referenced by CF_TableInit().

Here is the call graph for this function:



12.19.4 Variable Documentation

12.19.4.1 CF_AppData `CF_AppData_t` CF_AppData

Singleton instance of the application global data.

Definition at line 40 of file cf_app.c.

Referenced by CF_AbandonCmd(), CF_AppInit(), CF_AppMain(), CF_AppPipe(), CF_CancelCmd(), CF_CFDP_AppendTlv(), CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), CF_CFDP_CycleEngine(), CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_HandleNotKeepFile(), CF_CFDP_InitEngine(), CF_CFDP_IsPollingDir(), CF_CFDP_MsgOutGet(), CF_CFDP_PlaybackDir(), CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_DispatchRecv(), CF_CFDP_R_Init(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), CF_CFDP_RecvDrop(), CF_CFDP_RecvFd(), CF_CFDP_RecvIdle(), CF_CFDP_RecvMd(), CF_CFDP_RecvPh(), CF_CFDP_ResetTransaction(), CF_CFDP_S2_Nak(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_DispatchRecv(), CF_CFDP_S_SendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_CFDP_S_Tick(), CF_CFDP_Send(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), CF_CFDP_TxFile(), CF_CFDP_TxFile_Initiate(), CF_CheckTables(), CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_DisableEngineCmd(), CF_DoEnableDisableDequeue(), CF_DoEnableDisablePolldir(), CF_DoFreezeThaw(), CF_DoPurgeQueue(), CF_DoSuspRes(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_EnableEngineCmd(), CF_FindTransactionBySequenceNumberAllChannels(), CF_FreeTransaction(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_InsertSortPrio(), CF_MoveTransaction(), CF_NoopCmd(), CF_PlaybackDirCmd(), CF_ProcessGroundCommand(), CF_PurgeQueueCmd(), CF_ResetCountersCmd(), CF_SendHkCmd(), CF_TableInit(), CF_ThawCmd(), CF_Timer_Sec2Ticks(), CF_TraverseAllTransactions_All_Channels(), CF_TsnChanAction(), CF_TxFileCmd(), CF_ValidateMaxOutgoingCmd(), and CF_WriteQueueCmd().

12.20 apps/cf/fsw/src(cf_assert.h File Reference

```
#include "cfe.h"
```

Macros

CF assert macro

CF Assert statements within the code are primarily informational for developers, as the conditions within them should always be true. Barring any unforeseen bugs in the code, they should never get triggered. However, if the code is modified, these conditions could happen, so it is still worthwhile to keep these statements in the source code, so they can be enabled if necessary.

The debug build assert translates CF Assert to the system assert. Note that asserts may still get disabled if building with NDEBUG flag set, even if CF_DEBUG_BUILD flag is enabled.

It should be impossible to get any conditions which are asserted, so it should be safe to turn these off via the normal build assert. This is the configuration that the code should be normally tested and verified in.

- #define CF Assert(x) /* no-op */
Normal build assert.

12.20.1 Detailed Description

The CF Application CF Assert macro

12.20.2 Macro Definition Documentation

12.20.2.1 CF Assert #define CF Assert (x) /* no-op */

Normal build assert.

Definition at line 63 of file cf_assert.h.

12.21 apps/cf/fsw/src/cf_cfdp.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_dispatch.h"
#include "cf_cfdp_sbintf.h"
#include <string.h>
#include "cf_assert.h"
```

Functions

- void **CF_CFDP_EncodeStart** (CF_EncoderState_t *penc, void *msgbuf, CF_Logical_PduBuffer_t *ph, size_t encaps_hdr_size, size_t total_size)
Initiate the process of encoding a new PDU to send.
- void **CF_CFDP_DecodeStart** (CF_DecoderState_t *pdec, const void *msgbuf, CF_Logical_PduBuffer_t *ph, size_t encaps_hdr_size, size_t total_size)
Initiate the process of decoding a received PDU.
- void **CF_CFDP_ArmAckTimer** (CF_Transaction_t *txn)
Arm the ACK timer.
- static CF_CFDP_Class_t **CF_CFDP_GetClass** (const CF_Transaction_t *txn)
- static bool **CF_CFDP_IsSender** (CF_Transaction_t *txn)
- static void **CF_CFDP_ArmInactTimer** (CF_Transaction_t *txn)
- void **CF_CFDP_DispatchRecv** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Dispatch received packet to its handler.

- static void `CF_CFDP_DispatchTx` (`CF_Transaction_t` *`txn`)
- static `CF_ChunkWrapper_t` * `CF_CFDP_FindUnusedChunks` (`CF_Channel_t` *`chan`, `CF_Direction_t` `dir`)
- static void `CF_CFDP_SetPduLength` (`CF_Logical_PduBuffer_t` *`ph`)
- `CF_Logical_PduBuffer_t` * `CF_CFDP_ConstructPduHeader` (const `CF_Transaction_t` *`txn`, `CF_CFDP_FileDirective_t` `directive_code`, `CF_EntityId_t` `src_eid`, `CF_EntityId_t` `dst_eid`, bool `towards_sender`, `CF_TransactionSeq_t` `tsn`, bool `silent`)
Build the PDU header in the output buffer to prepare to send a packet.
- `CFE_Status_t` `CF_CFDP_SendMd` (`CF_Transaction_t` *`txn`)
Build a metadata PDU for transmit.
- `CFE_Status_t` `CF_CFDP_SendFd` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Send a previously-assembled filedata PDU for transmit.
- void `CF_CFDP_AppendTlv` (`CF_Logical_TlvList_t` *`ptlv_list`, `CF_CFDP_TlvType_t` `tlv_type`)
Appends a single TLV value to the logical PDU data.
- `CFE_Status_t` `CF_CFDP_SendEof` (`CF_Transaction_t` *`txn`)
Build an EOF PDU for transmit.
- `CFE_Status_t` `CF_CFDP_SendAck` (`CF_Transaction_t` *`txn`, `CF_CFDP_AckTxnStatus_t` `ts`, `CF_CFDP_FileDirective_t` `dir_code`, `CF_CFDP_ConditionCode_t` `cc`, `CF_EntityId_t` `peer_eid`, `CF_TransactionSeq_t` `tsn`)
Build an ACK PDU for transmit.
- `CFE_Status_t` `CF_CFDP_SendFin` (`CF_Transaction_t` *`txn`, `CF_CFDP_FinDeliveryCode_t` `dc`, `CF_CFDP_FinFileStatus_t` `fs`, `CF_CFDP_ConditionCode_t` `cc`)
Build a FIN PDU for transmit.
- `CFE_Status_t` `CF_CFDP_SendNak` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Send a previously-assembled NAK PDU for transmit.
- `CFE_Status_t` `CF_CFDP_RecvPh` (`uint8 chan_num`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack a basic PDU header from a received message.
- `CFE_Status_t` `CF_CFDP_RecvMd` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack a metadata PDU from a received message.
- `CFE_Status_t` `CF_CFDP_RecvFd` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack a file data PDU from a received message.
- `CFE_Status_t` `CF_CFDP_RecvEof` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack an EOF PDU from a received message.
- `CFE_Status_t` `CF_CFDP_RecvAck` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack an ACK PDU from a received message.
- `CFE_Status_t` `CF_CFDP_RecvFin` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack an FIN PDU from a received message.
- `CFE_Status_t` `CF_CFDP_RecvNak` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Unpack a NAK PDU from a received message.
- void `CF_CFDP_RecvDrop` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Receive state function to ignore a packet.
- void `CF_CFDP_RecvIdle` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Receive state function to process new rx transaction.
- `CFE_Status_t` `CF_CFDP_InitEngine` (void)
Initialization function for the CFDP engine.
- `CF_CListTraverse_Status_t` `CF_CFDP_CycleTxFirstActive` (`CF_CListNode_t` *`node`, void *`context`)
List traversal function that cycles the first active tx.
- void `CF_CFDP_CycleTx` (`CF_Channel_t` *`chan`)
Cycle the current active tx or make a new one active.
- `CF_CListTraverse_Status_t` `CF_CFDP_DoTick` (`CF_CListNode_t` *`node`, void *`context`)

- *List traversal function that calls a r or s tick function.*
- void [CF_CFDP_TickTransactions](#) (CF_Channel_t *chan)
 - Call R and then S tick functions for all active transactions.*
- void [CF_CFDP_InitTxnTxFile](#) (CF_Transaction_t *txn, CF_CFDP_Class_t cfdp_class, uint8 keep, uint8 chan, uint8 priority)
 - Helper function to set tx file state in a transaction.*
- static void [CF_CFDP_TxFile_Initiate](#) (CF_Transaction_t *txn, CF_CFDP_Class_t cfdp_class, uint8 keep, uint8 chan, uint8 priority, CF_EntityId_t dest_id)
 - Begin transmit of a file.*
- CFE_Status_t [CF_CFDP_TxFile](#) (const char *src_filename, const char *dst_filename, CF_CFDP_Class_t cfdp_class, uint8 keep, uint8 chan_num, uint8 priority, CF_EntityId_t dest_id)
 - Begin transmit of a directory.*
- void [CF_CFDP_ProcessPlaybackDirectory](#) (CF_Channel_t *chan, CF_Playback_t *pb)
 - Step each active playback directory.*
- static void [CF_CFDP_UpdatePollPbCounted](#) (CF_Playback_t *pb, int up, uint8 *counter)
 - Kick the dir playback if timer elapsed.*
- static void [CF_CFDP_ProcessPlaybackDirectories](#) (CF_Channel_t *chan)
 - void [CF_CFDP_ProcessPollingDirectories](#) (CF_Channel_t *chan)*
- void [CF_CFDP_ProcessPollingDirectories](#) (CF_Channel_t *chan)
 - Cycle the engine. Called once per wakeup.*
- void [CF_CFDP_ResetTransaction](#) (CF_Transaction_t *txn, bool keep_history)
 - Reset a transaction and all its internals to an unused state.*
- void [CF_CFDP_SetTxnStatus](#) (CF_Transaction_t *txn, CF_TxnStatus_t txn_stat)
 - Helper function to store transaction status code only.*
- void [CF_CFDP_SendEotPkt](#) (CF_Transaction_t *txn)
 - Send an end of transaction packet.*
- int [CF_CFDP_CopyStringFromLV](#) (char *buf, size_t buf_maxsz, const CF_Logical_Lv_t *src_lv)
 - Copy string data from a lv (length, value) pair.*
- void [CF_CFDP_CancelTransaction](#) (CF_Transaction_t *txn)
 - Cancels a transaction.*
- CF_CListTraverse_Status_t [CF_CFDP_CloseFiles](#) (CF_CListNode_t *node, void *context)
 - List traversal function to close all files in all active transactions.*
- void [CF_CFDP_DisableEngine](#) (void)
 - Disables the CFDP engine and resets all state in it.*
- bool [CF_CFDP_IsPollingDir](#) (const char *src_file, uint8 chan_num)
 - Check if source file came from polling directory.*
- void [CF_CFDP_HandleNotKeepFile](#) (CF_Transaction_t *txn)
 - Remove/Move file after transaction.*
- void [CF_CFDP_MoveFile](#) (const char *src, const char *dest_dir)
 - Move File.*

12.21.1 Detailed Description

The CF Application main CFDP engine and PDU parsing implementation

This file contains two sets of functions. The first is what is needed to deal with CFDP PDUs. Specifically validating them for correctness and ensuring the byte-order is correct for the target. The second is incoming and outgoing CFDP PDUs pass through here. All receive CFDP PDU logic is performed here and the data is passed to the R (rx) and S (tx) logic.

12.21.2 Function Documentation

12.21.2.1 CF_CFDP_AppendTlv() void CF_CFDP_AppendTlv (

<code>CF_Logical_TlvList_t * ptlv_list,</code>
<code>CF_CFDP_TlvType_t tlv_type)</code>

Appends a single TLV value to the logical PDU data.

This function implements common functionality between SendEof and SendFin which append a TLV value specifying the faulting entity ID.

Assumptions, External Events, and Notes:

`ptlv_list` must not be NULL. Only `CF_CFDP_TLV_TYPE_ENTITY_ID` type is currently implemented

Parameters

<code>ptlv_list</code>	TLV list from current PDU buffer.
<code>tlv_type</code>	Type of TLV to append. Currently must be <code>CF_CFDP_TLV_TYPE_ENTITY_ID</code> .

Definition at line 371 of file cf_cfdp.c.

References `CF_AppData`, `CF_CFDP_GetValueEncodedSize()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_PDU_MAJ_X_TLV`, `CF_AppData_t::config_table`, `CF_Logical_Tlv::data`, `CF_Logical_TlvData::data_ptr`, `CF_Logical_TlvData::eid`, `CF_Logical_Tlv::length`, `CF_ConfigTable::local_eid`, `CF_Logical_TlvList::num_tlv`, `CF_Logical_TlvList::tlv`, and `CF_Logical_Tlv::type`.

Referenced by `CF_CFDP_SendEof()`, and `CF_CFDP_SendFin()`.

Here is the call graph for this function:



12.21.2.2 CF_CFDP_ArmAckTimer() void CF_CFDP_ArmAckTimer (

<code>CF_Transaction_t * txn)</code>

Arm the ACK timer.

Description

Helper function to arm the ACK timer and set the flag.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

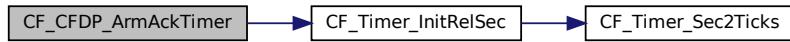
<code>txn</code>	Pointer to the transaction state
------------------	----------------------------------

Definition at line 123 of file cf_cfdp.c.

References CF_Transaction::ack_timer, CF_Flags_Common::ack_timer_armed, CF_ChannelConfig::ack_timer_s, CF_AppData, CF_Timer_InitRelSec(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_StateFlags::com, CF_AppData_t::config_table, and CF_Transaction::flags.

Referenced by CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_Init(), CF_CFDP_R_Tick(), CF_CFDP_S2_Nak_Arm(), and CF_CFDP_S_Tick().

Here is the call graph for this function:



12.21.2.3 CF_CFDP_ArmInactTimer() static void CF_CFDP_ArmInactTimer (

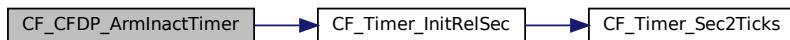
CF_Transaction_t * txn) [inline], [static]

Definition at line 158 of file cf_cfdp.c.

References CF_AppData, CF_Timer_InitRelSec(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Transaction::inactivity_timer, and CF_ChannelConfig::inactivity_timer_s.

Referenced by CF_CFDP_DispatchRecv(), and CF_CFDP_TxFile_Initiate().

Here is the call graph for this function:



12.21.2.4 CF_CFDP_CancelTransaction() void CF_CFDP_CancelTransaction (

CF_Transaction_t * txn)

Cancels a transaction.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

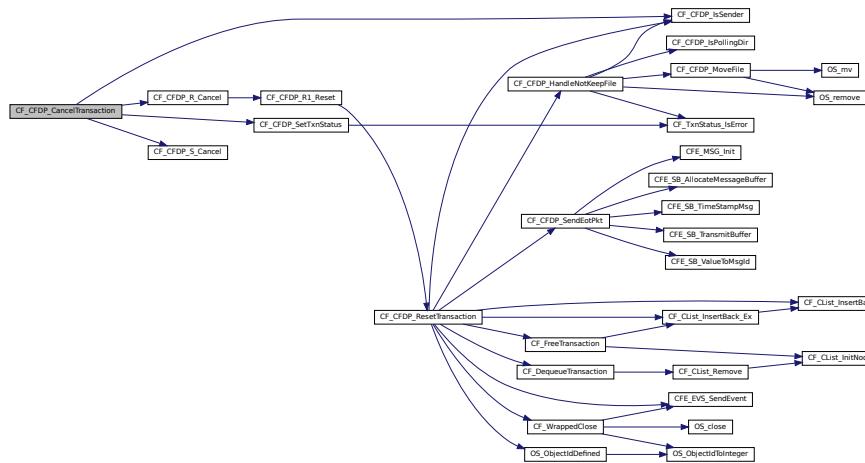
txn	Pointer to the transaction state
-----	----------------------------------

Definition at line 1734 of file cf_cfdp.c.

References CF_Flags_Common::canceled, CF_CFDP_IsSender(), CF_CFDP_R_Cancel(), CF_CFDP_S_Cancel(), CF_CFDP_SetTxnStatus(), CF_TxnStatus_CANCEL_REQUEST_RECEIVED, CF_StateFlags::com, and CF_Transaction::flags.

Referenced by CF_Cancel_TxnCmd().

Here is the call graph for this function:



12.21.2.5 CF_CFDP_CloseFiles() `CF_CListTraverse_Status_t CF_CFDP_CloseFiles (`
`CF_CListNode_t * node,`
`void * context)`

List traversal function to close all files in all active transactions.

This helper is used in conjunction with [CF_CList_Traverse\(\)](#).

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<code>node</code>	List node pointer
<code>context</code>	Opaque pointer, not used in this function

Returns

integer traversal code

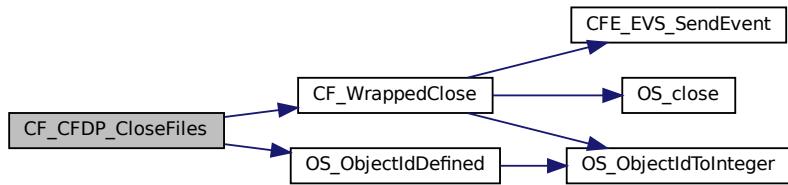
Return values

<code>Always</code>	CF_LIST_CONT indicate list traversal should not exit early.
---------------------	---

Definition at line 1751 of file cf_cfdp.c.

References CF_CLIST_CONT, CF_WrappedClose(), container_of, CF_Transaction::fd, and OS_ObjectIdDefined().
Referenced by CF_CFDP_DisableEngine().

Here is the call graph for this function:



```

12.21.2.6 CF_CFDP_ConstructPduHeader() CF_Logical_PduBuffer_t* CF_CFDP_ConstructPduHeader (
    const CF_Transaction_t * txn,
    CF_CFDP_FileDirective_t directive_code,
    CF_EntityId_t src_eid,
    CF_EntityId_t dst_eid,
    bool towards_sender,
    CF_TransactionSeq_t tsn,
    bool silent )
  
```

Build the PDU header in the output buffer to prepare to send a packet.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>directive_code</i>	Code to use for file directive headers (set to 0 for data)
<i>src_eid</i>	Value to set in source entity ID field
<i>dst_eid</i>	Value to set in destination entity ID field
<i>towards_sender</i>	Whether this is transmitting toward the sender entity
<i>tsn</i>	Transaction sequence number to put into PDU
<i>silent</i>	If true, suppress error event if no message buffer available

Returns

Pointer to PDU buffer which may be filled with additional data

Return values

<i>NULL</i>	if no message buffer available
-------------	--------------------------------

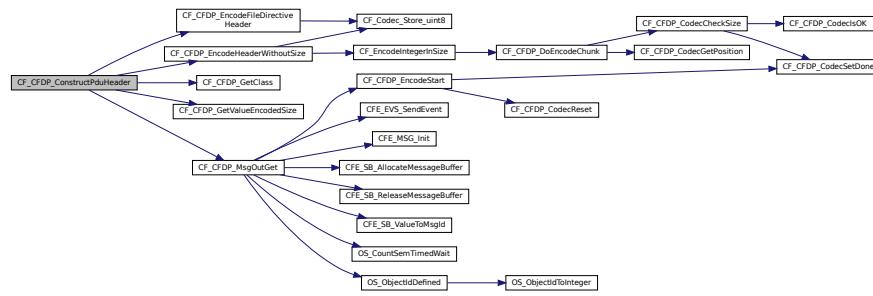
Definition at line 238 of file cf_cfdp.c.

References CF_CFDP_CLASS_1, CF_CFDP_EncodeFileDirectiveHeader(), CF_CFDP_EncodeHeaderWithoutSize(), CF_CFDP_GetClass(), CF_CFDP_GetValueEncodedSize(), CF_CFDP_MsgOutGet(), CF_Logical_Pdu

Header::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduFileDirectiveHeader::directive_code, CF_Logical_PduHeader::eid_length, CF_Logical_PduBuffer::fdirective, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_Logical_PduBuffer::penc, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_R_SubstateSendNak(), CF_CFDP_S_SendFileData(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), and CF_CFDP_SendMd().

Here is the call graph for this function:



12.21.2.7 CF_CFDP_CopyStringFromLV()

```
int CF_CFDP_CopyStringFromLV (
    char * buf,
    size_t buf_maxsz,
    const CF_Logical_Lv_t * src_lv )
```

Copy string data from a lv (length, value) pair.

This copies a string value from an LV pair inside a PDU buffer. In CF this is used for file names embedded within PDUs.

Note

This function assures that the output string is terminated appropriately, such that it can be used as a normal C string. As such, the buffer size must be at least 1 byte larger than the maximum string length.

Assumptions, External Events, and Notes:

src_lv must not be NULL. buf must not be NULL.

Parameters

<i>buf</i>	Pointer to buffer to store string
<i>buf_maxsz</i>	Total size of buffer pointer to by buf (usable size is 1 byte less, for termination)
<i>src_lv</i>	Pointer to LV pair from logical PDU buffer

Returns

The resulting string length, NOT including termination character

Return values

CF_ERROR	on error
----------	----------

Definition at line 1714 of file cf_cfdp.c.

References CF_ERROR, CF_Logical_Lv::data_ptr, and CF_Logical_Lv::length.

Referenced by CF_CFDP_RecvMd().

12.21.2.8 CF_CFDP_CycleEngine()

```
void CF_CFDP_CycleEngine (
```

```
    void )
```

Cycle the engine. Called once per wakeup.

Assumptions, External Events, and Notes:

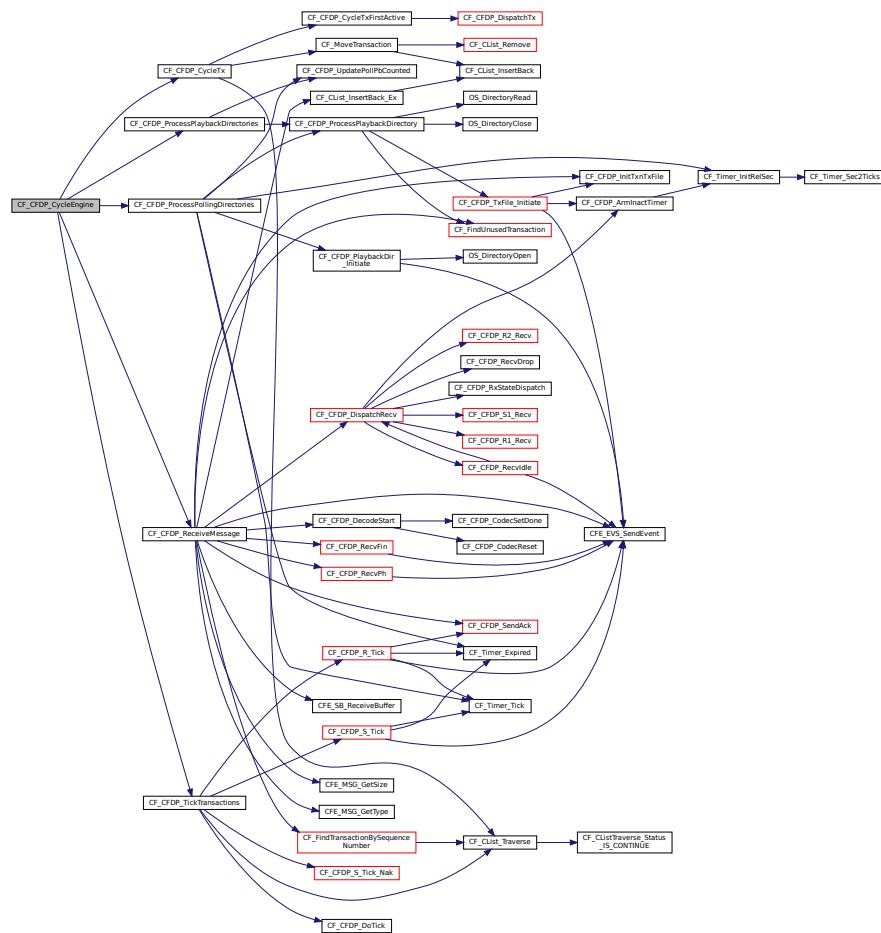
None

Definition at line 1549 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CycleTx(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_ReceiveMessage(), CF_CFDP_TickTransactions(), CF_NUM_CHANNELS, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Engine::enabled, CF_AppData_t::engine, CF_HkChannel_Data::frozen, CF_AppData_t::hk, CF_Engine::outgoing_counter, and CF_HkPacket::Payload.

Referenced by CF_WakeupCmd().

Here is the call graph for this function:



12.21.2.9 CF_CFDP_CycleTx() `void CF_CFDP_CycleTx (`
`CF_Channel_t * chan)`

Cycle the current active tx or make a new one active.

Description

First traverses all tx transactions on the active queue. If at least one is found, then it stops. Otherwise it moves a transaction on the pending queue to the active queue and tries again to find an active one.

Assumptions, External Events, and Notes:

None

Parameters

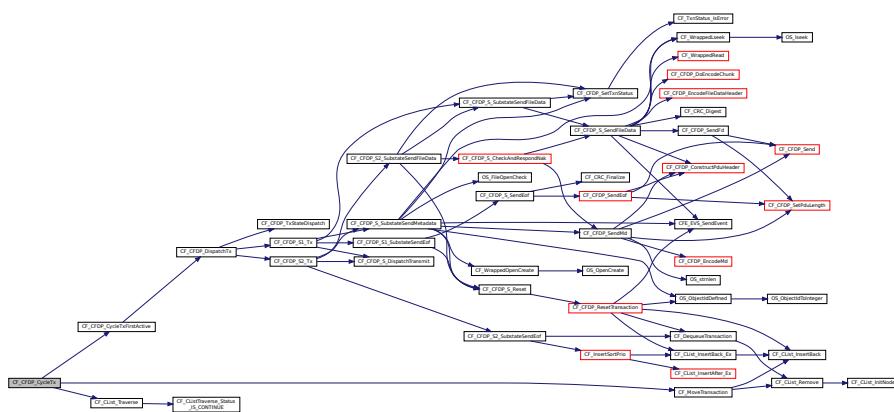
<code>chan</code>	Channel to cycle
-------------------	------------------

Definition at line 1077 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CycleTxFirstActive(), CF_CList_Traverse(), CF_MoveTransaction(), CF_QueueIdx_PEND, CF_QueueIdx_TXA, CF_ConfigTable::chan, CF_Engine::channels, CF_AppData_t::config_table, container_of, CF_Channel::cur, CF_ChannelConfig::dequeue_enabled, CF_AppData_t::engine, CF_Channel::qs, and CF_CFDP_CycleTx_args::ran_one.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.21.2.10 CF_CFDP_CycleTxFirstActive() `CF_CListTraverse_Status_t CF_CFDP_CycleTxFirstActive (`
`CF_CListNode_t * node,`
`void * context)`

List traversal function that cycles the first active tx.

This helper is used in conjunction with [CF_CList_Traverse\(\)](#).

Description

There can only be one active tx transaction per engine cycle. This function finds the first active, and then sends file data PDUs until there are no outgoing message buffers.

Assumptions, External Events, and Notes:

node must not be NULL. Context must not be NULL.

Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_CycleTx_args_t object (passed through)

Returns

integer traversal code

Return values

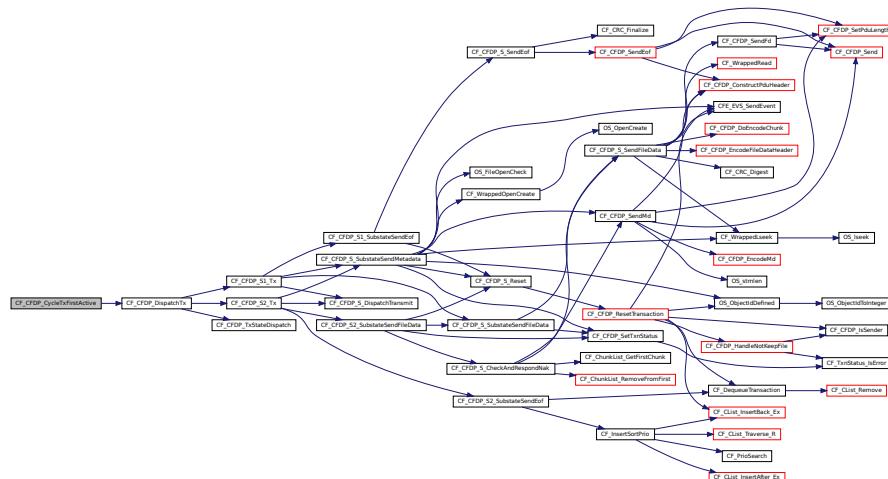
<code>CF_CLIST_EXIT</code>	when it's found, which terminates list traversal
<code>CF_CLIST_CONT</code>	when it's isn't found, which causes list traversal to continue

Definition at line 1041 of file cf_cfdp.c.

References CF_ASSERT, CF_CFDP_DispatchTx(), CF_CLIST_CONT, CF_CLIST_EXIT, CF_PERF_ID_PDUSENT, CF_QueueIdx_TXA, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CF_CFDP_CycleTx_args::chan, CF_Transaction::chan_num, CF_StateFlags::com, container_of, CF_Channel::cur, CF_Transaction::flags, CF_Flags_Common::q_index, CF_CFDP_CycleTx_args::ran_one, and CF_Flags_Common::suspended.

Referenced by CF_CFDP_CycleTx().

Here is the call graph for this function:



12.21.2.11 CF_CFDP_DecodeStart() void CF_CFDP_DecodeStart (

```
CF_DecoderState_t * pdec,  
const void * msgbuf,  
CF_Logical_PduBuffer_t * ph,  
size_t encaps_hdr_size,  
size_t total_size )
```

Initiate the process of decoding a received PDU.

This resets the decoder and PDU buffer to initial values, and prepares for decoding a new PDU that was received from a remote entity.

Parameters

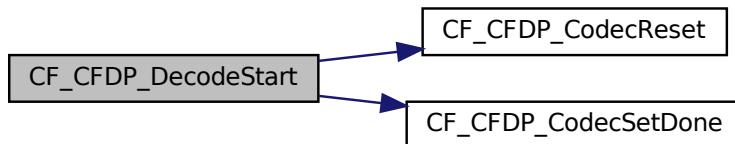
<i>pdec</i>	Decoder state structure, will be reset-initialized by this call to point to msgbuf.
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Total size of msgbuf encapsulation structure (decoding cannot exceed this)

Definition at line 89 of file cf_cfdp.c.

References CF_DecoderState::base, CF_CFDP_CodecReset(), CF_CFDP_CodecSetDone(), CF_DecoderState::codec_state, CF_CodecState::max_size, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



12.21.2.12 CF_CFDP_DisableEngine() void CF_CFDP_DisableEngine (void)

Disables the CFDP engine and resets all state in it.

Assumptions, External Events, and Notes:

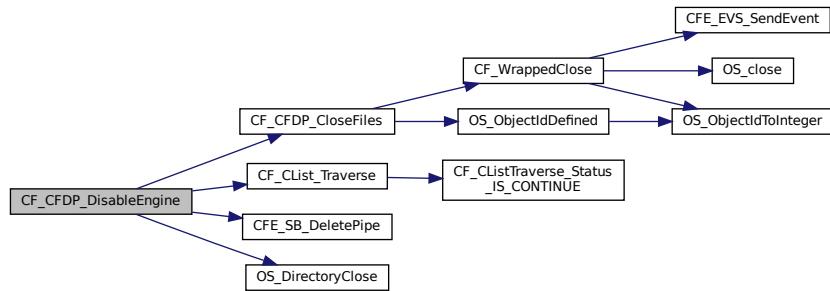
None

Definition at line 1767 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_CloseFiles(), CF_CList_Traverse(), CF_MAX_COMMAND_ED_PLAYBACK_DIRECTORIES_PER_CHAN, CF_MAX_POLLING_DIR_PER_CHAN, CF_NUM_CHANNELS, CF_QueueIdx_RX, CF_QueueIdx_TXA, CF_QueueIdx_TXW, CFE_SB_DeletePipe(), CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Playback::dir_id, CF_Engine::enabled, CF_AppData_t::engine, CF_AppData_t::hk, OS_DirectoryClose(), CF_HkPacket::Payload, CF_Poll::pb, CF_Channel::pipe, CF_Channel::playback, CF_Channel::poll, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_DisableEngineCmd().

Here is the call graph for this function:



12.21.2.13 CF_CFDP_DispatchRecv() void CF_CFDP_DispatchRecv (CF_Transaction_t * *txn*, CF_Logical_PduBuffer_t * *ph*)

Dispatch received packet to its handler.

This dispatches the PDU to the appropriate handler based on the transaction state

Assumptions, External Events, and Notes:

txn must not be null. It must be an initialized transaction.

Parameters

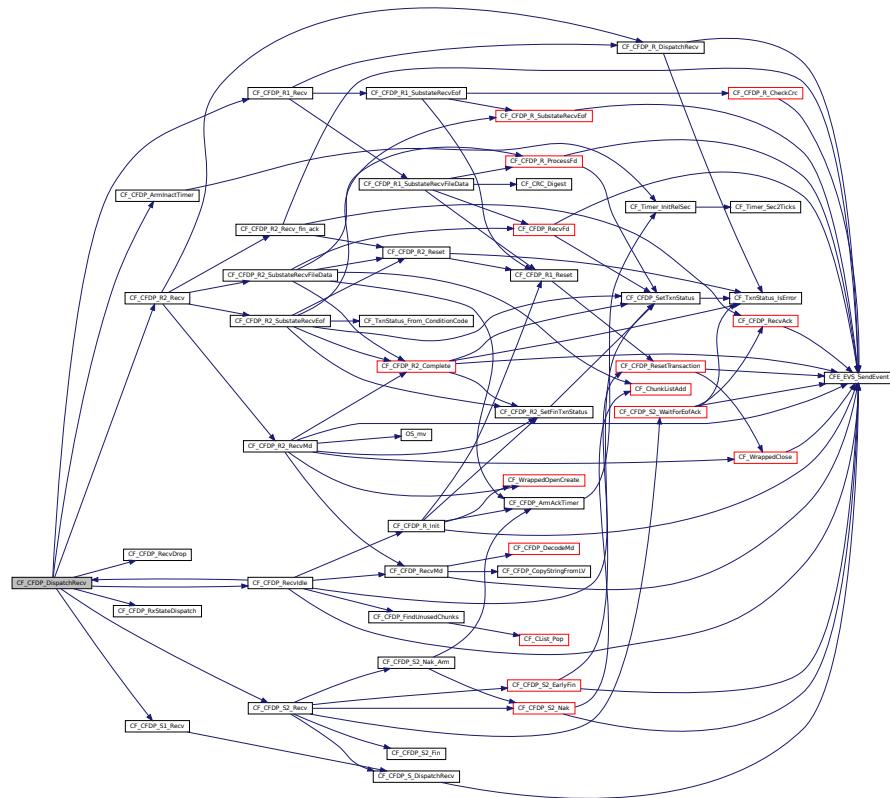
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 169 of file cf_cfdp.c.

References CF_CFDP_ArmInactTimer(), CF_CFDP_R1_Recv(), CF_CFDP_R2_Recv(), CF_CFDP_RecvDrop(), CF_CFDP_RecvIdle(), CF_CFDP_RxStateDispatch(), CF_CFDP_S1_Recv(), CF_CFDP_S2_Recv(), CF_TxnState_DR_OP, CF_TxnState_IDLE, CF_TxnState_R1, CF_TxnState_R2, CF_TxnState_S1, CF_TxnState_S2, and CF_CFDP_TxnRecvDispatchTable_t::rx.

Referenced by CF_CFDP_ReceiveMessage(), and CF_CFDP_RecvIdle().

Here is the call graph for this function:



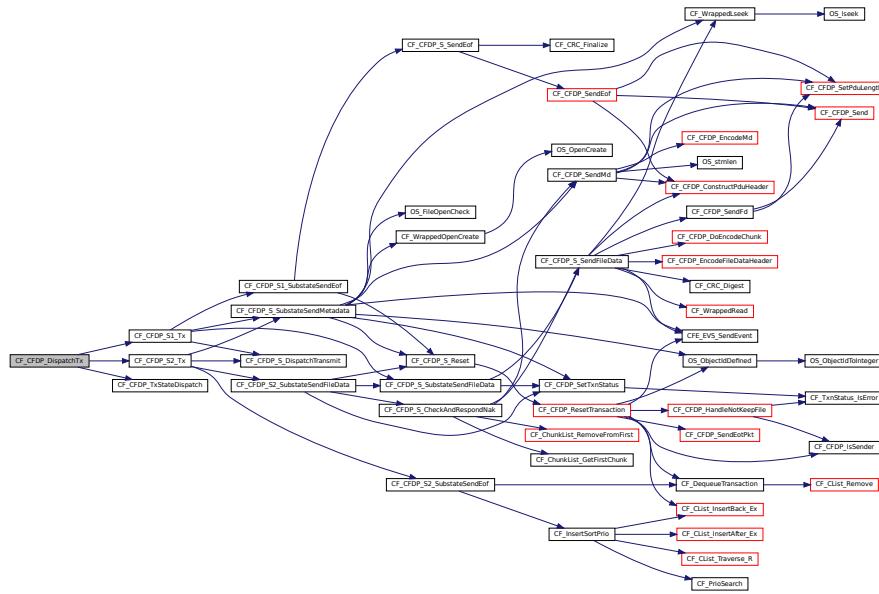
12.21.2.14 CF_CFDP_DispatchTx() static void CF_CFDP_DispatchTx (CF_Transaction_t * txn) [static]

Definition at line 187 of file cf_cfdp.c.

References CF_CFDP_S1_Tx(), CF_CFDP_S2_Tx(), State S2, and CF_CFDP_TxnSendDispatchTable t::tx.

Referenced by CF_CFDP_CycleTxFirstActive().

Here is the call graph for this function:



```
12.21.2.15 CF_CFDP_DoTick() CF_CListTraverse_Status_t CF_CFDP_DoTick (
    CF_CListNode_t * node,
    void * context )
```

List traversal function that calls a r or s tick function.

This helper is used in conjunction with [CF_CList_Traverse\(\)](#).

Assumptions, External Events, and Notes:

node must not be NULL, context must not be NULL.

Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_Tick_args_t object (passed through)

Returns

integer traversal code

Return values

<i>CF_CLIST_EXIT</i>	when it's found, which terminates list traversal
<i>CF_CLIST_CONT</i>	when it's isn't found, which causes list traversal to continue

Definition at line 1120 of file cf_cfdp.c.

References CF_CLIST_CONT, CF_CLIST_EXIT, CF_CFDP_Tick_args::chan, CF_StateFlags::com, CF_CFDP_Tick→

_args::cont, container_of, CF_Channel::cur, CF_CFDP_Tick_args::early_exit, CF_Transaction::flags, CF_CFDP_Tick_args::fn, and CF_Flags_Common::suspended.
Referenced by CF_CFDP_TickTransactions().

12.21.2.16 CF_CFDP_EncodeStart() void CF_CFDP_EncodeStart (

```
CF_EncoderState_t * penc,
void * msgbuf,
CF_Logical_PduBuffer_t * ph,
size_t encaps_hdr_size,
size_t total_size )
```

Initiate the process of encoding a new PDU to send.

This resets the encoder and PDU buffer to initial values, and prepares for encoding a new PDU for sending to a remote entity.

Parameters

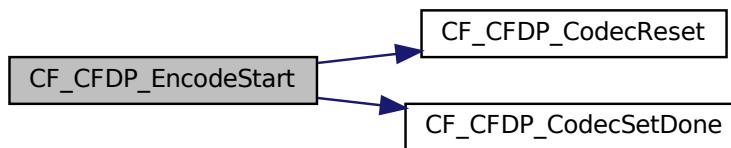
<i>penc</i>	Encoder state structure, will be reset-initialized by this call to point to msgbuf.
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Allocated size of msgbuf encapsulation structure (encoding cannot exceed this)

Definition at line 55 of file cf_cfdp.c.

References CF_EncoderState::base, CF_CFDP_CodecReset(), CF_CFDP_CodecSetDone(), CF_EncoderState::codec_state, CF_CodecState::max_size, and CF_Logical_PduBuffer::penc.

Referenced by CF_CFDP_MsgOutGet().

Here is the call graph for this function:



12.21.2.17 CF_CFDP_FindUnusedChunks() static CF_ChunkWrapper_t* CF_CFDP_FindUnusedChunks (

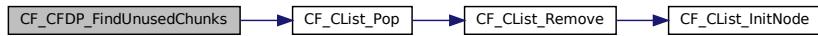
```
CF_Channel_t * chan,
CF_Direction_t dir ) [static]
```

Definition at line 200 of file cf_cfdp.c.

References CF_ASSERT, CF_CList_Pop(), CF_Direction_NUM, container_of, and CF_Channel::cs.

Referenced by CF_CFDP_RecvIdle(), and CF_CFDP_TxFile_Initiate().

Here is the call graph for this function:



12.21.2.18 CF_CFDP_GetClass() static CF_CFDP_Class_t CF_CFDP_GetClass (const CF_Transaction_t * txn) [inline], [static]

Definition at line 134 of file cf_cfdp.c.

References CF_ASSERT, CF_QueueIdx_FREE, CF_TxnState_R2, CF_TxnState_S2, CF_StateFlags::com, CF_Transaction::flags, CF_Flags_Common::q_index, and CF_Transaction::state.

Referenced by CF_CFDP_ConstructPduHeader(), and CF_CFDP_SendNak().

12.21.2.19 CF_CFDP_HandleNotKeepFile() void CF_CFDP_HandleNotKeepFile (CF_Transaction_t * txn)

Remove/Move file after transaction.

This helper is used to handle "not keep" file option after a transaction.

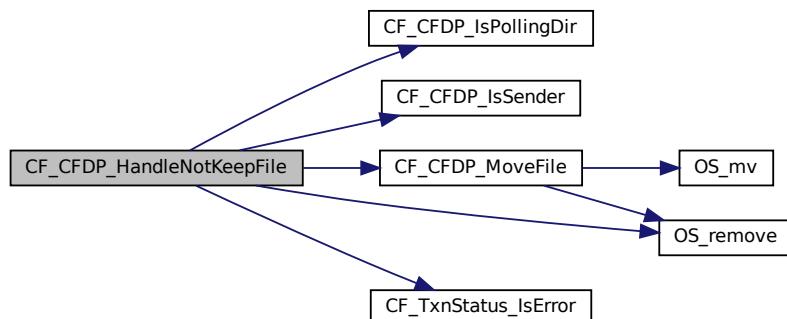
Assumptions, External Events, and Notes:

Definition at line 1850 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_IsPollingDir(), CF_CFDP_IsSender(), CF_CFDP_MoveFile(), CF_TxnStatus_IsError(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_TxnFilenames::dst_filename, CF_ConfigTable::fail_dir, CF_History::fnames, CF_Transaction::history, CF_ChannelConfig::move_dir, OS_remove(), CF_TxnFilenames::src_filename, and CF_History::txn_stat.

Referenced by CF_CFDP_ResetTransaction().

Here is the call graph for this function:



12.21.2.20 CF_CFDP_InitEngine() `CFE_Status_t` CF_CFDP_InitEngine (
 `void`)

Initialization function for the CFDP engine.

Description

Performs all initialization of the CFDP engine

Assumptions, External Events, and Notes:

Only called once.

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
--------------------------	--

Returns

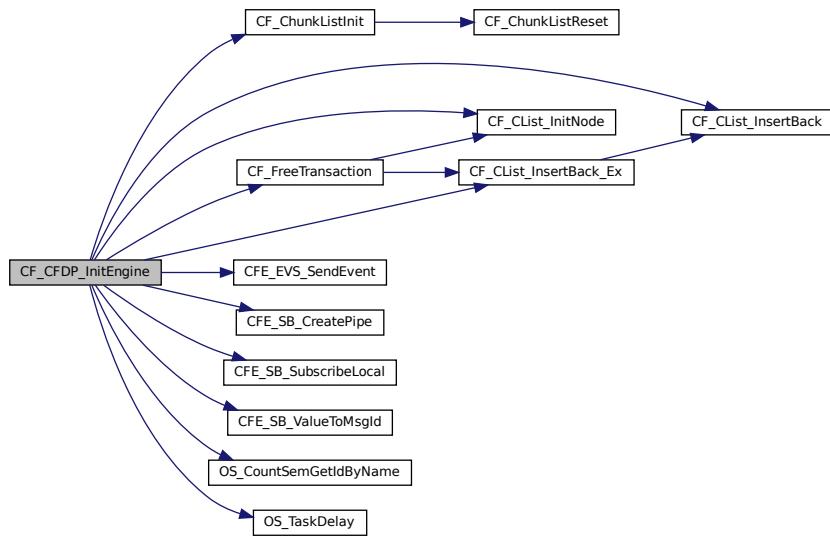
anything else on error.

Definition at line 931 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION, CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION, CF_CHANNEL_PIPE_PREFIX, CF_ChunkListInit(), CF_CList_InitNode(), CF_CList_InsertBack(), CF_CList_InsertBack_Ex(), CF_CR_CHANNEL_PIPE_ERR_EID, CF_Direction_NUM, CF_FreeTransaction(), CF_INIT_SEM_ERR_EID, CF_INIT_SUB_ERR_EID, CF_NUM_CHANNELS, CF_NUM_CHANNELS_ALL_CHANNELS, CF_NUM_HISTORIES_PER_CHANNEL, CF_NUM_TRANSACTIONS_PER_CHANNEL, CF_QueueIdx_HIST_FREE, CF_STARTUP_SEM_MAX_RETRIES, CF_STARTUP_SEM_TASK_DELAY, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_CreatePipe(), CFE_SB_SubscribeLocal(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CF_ConfigTable::chan, CF_Transaction::chan_num, CF_Engine::channels, CF_Engine::chunk_mem, CF_ChunkWrapper::chunks, CF_Engine::chunks, CF_History::cl_node, CF_ChunkWrapper::cl_node, CF_AppData_t::config_table, CF_Channel::cs, CF_Engine::enabled, CF_AppData_t::engine, CF_Engine::histories, CF_ChannelConfig::mid_input, OS_CountSemGetIdByName(), OS_ERR_NAME_NOT_FOUND, OS_SUCCESS, OS_TaskDelay(), CF_Channel::pipe, CF_ChannelConfig::pipe_depth_input, CF_Channel::sem_id, CF_ChannelConfig::sem_name, and CF_Engine::transactions.

Referenced by CF_AppInit(), and CF_EnableEngineCmd().

Here is the call graph for this function:



```
12.21.2.21 CF_CFDP_InitTxnTxFile() void CF_CFDP_InitTxnTxFile (
    CF_Transaction_t * txn,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority )
```

Helper function to set tx file state in a transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for sending a file.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>cfdp_class</i>	Set to class 1 or class 2
<i>keep</i>	Whether to keep the local file
<i>chan</i>	CF channel number
<i>priority</i>	Priority of transfer

Definition at line 1216 of file cf_cfdp.c.

References CF_TxnState_S1, CF_TxnState_S2, CF_Transaction::chan_num, CF_Transaction::keep, CF_Transaction::priority, and CF_Transaction::state.

Referenced by CF_CFDP_ReceiveMessage(), and CF_CFDP_TxFile_Initiate().

```
12.21.2.22 CF_CFDP_IsPollingDir() bool CF_CFDP_IsPollingDir (
    const char * src_file,
    uint8 chan_num )
```

Check if source file came from polling directory.

Assumptions, External Events, and Notes:

Return values

true/false	<input type="button" value=""/>
------------	---------------------------------

Definition at line 1816 of file cf_cfdp.c.

References CF_AppData, CF_FILENAME_MAX_LEN, CF_MAX_POLLING_DIR_PER_CHAN, CF_ConfigTable::chan, CF_AppData_t::config_table, CF_ChannelConfig::polldir, and CF_PollDir::src_dir.

Referenced by CF_CFDP_HandleNotKeepFile().

```
12.21.2.23 CF_CFDP_IsSender() static bool CF_CFDP_IsSender (
    CF_Transaction_t * txn) [inline], [static]
```

Definition at line 145 of file cf_cfdp.c.

References CF_Assert, CF_QueueIdx_FREE, CF_TxnState_S1, CF_TxnState_S2, CF_StateFlags::com, CF_Transaction::flags, CF_Flags_Common::q_index, and CF_Transaction::state.

Referenced by CF_CFDP_CancelTransaction(), CF_CFDP_HandleNotKeepFile(), CF_CFDP_ResetTransaction(), and CF_CFDP_SendAck().

```
12.21.2.24 CF_CFDP_MoveFile() void CF_CFDP_MoveFile (
```

```
    const char * src,
    const char * dest_dir )
```

Move File.

This helper is used to move a file.

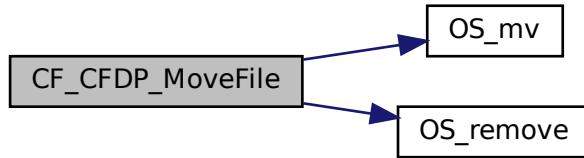
Assumptions, External Events, and Notes:

Definition at line 1884 of file cf_cfdp.c.

References OS_ERROR, OS_MAX_PATH_LEN, OS_mv(), OS_remove(), and OS_SUCCESS.

Referenced by CF_CFDP_HandleNotKeepFile().

Here is the call graph for this function:



12.21.2.25 CF_CFDP_PlaybackDir() CFE_Status_t CF_CFDP_PlaybackDir (

```

const char * src_filename,
const char * dst_filename,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
uint16 dest_id )
  
```

Begin transmit of a directory.

Description

This function sets up CF_Playback_t structure with state so it can become part of the directory polling done at each engine cycle.

Assumptions, External Events, and Notes:

src_filename must not be NULL. dst_filename must not be NULL.

Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
-----------------------------	--

Returns

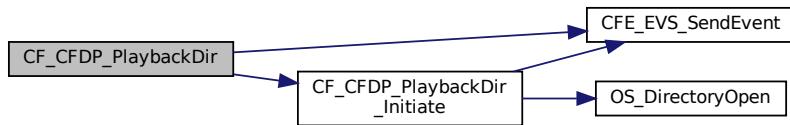
CFE_SUCCESS on success. CF_ERROR on error.

Definition at line 1343 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_DIR_SLOT_ERR_EID, CF_CFDP_PlaybackDir_Initiate(), CF_ERROR, CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Engine::channels, CF_AppData_t::engine, and CF_Channel::playback.

Referenced by CF_PlaybackDirCmd().

Here is the call graph for this function:



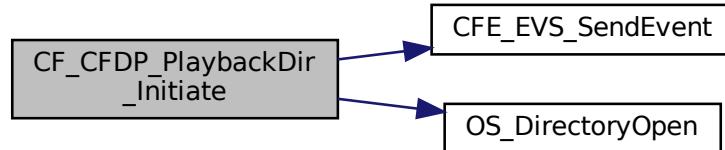
12.21.2.26 CF_CFDP_PlaybackDir_Initiate() static CFE_Status_t CF_CFDP_PlaybackDir_Initiate (CF_Playback_t * pb,
const char * src_filename,
const char * dst_filename,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
CF_EntityId_t dest_id) [static]

Definition at line 1303 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_OPENDIR_ERR_EID, CF_Playback::cfdp_class, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Playback::dest_id, CF_Playback::dir_id, CF_HkFault::directory_read, CF_Playback::diopen, CF_TxnFilenames::dst_filename, CF_HkCounters::fault, CF_Playback::fnames, CF_AppData_t::hk, CF_Playback::keep, OS_DirectoryOpen(), OS_SUCCESS, CF_HkPacket::Payload, CF_Playback::priority, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_PlaybackDir(), and CF_CFDP_ProcessPollingDirectories().

Here is the call graph for this function:



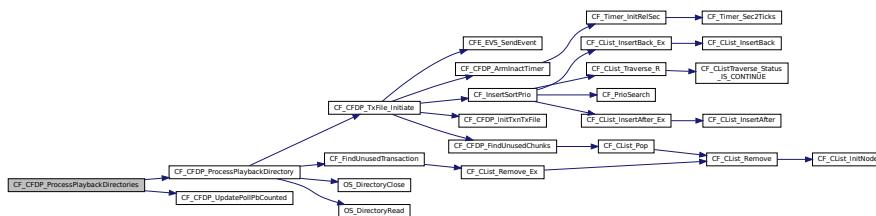
12.21.2.27 CF_CFDP_ProcessPlaybackDirectories() static void CF_CFDP_ProcessPlaybackDirectories (CF_Channel_t * chan) [static]

Definition at line 1461 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_UpdatePollPbCounted(), CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::engine, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Channel::playback, and CF_HkChannel_Data::playback_counter.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.21.2.28 CF_CFDP_ProcessPlaybackDirectory() void CF_CFDP_ProcessPlaybackDirectory (CF_Channel_t * chan, CF_Playback_t * pb)

Step each active playback directory.

Description

Check if a playback directory needs iterated, and if so does, and if a valid file is found initiates playback on it.

Assumptions, External Events, and Notes:

chan must not be NULL, pb must not be NULL.

Parameters

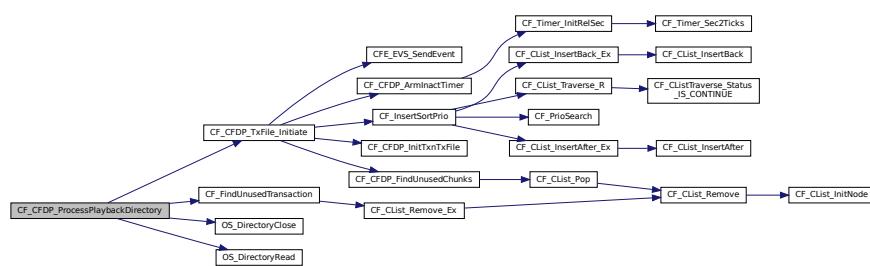
<i>chan</i>	The channel associated with the playback
<i>pb</i>	The playback state

Definition at line 1373 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_Assert, CF_CFDP_TxFile_Initiate(), CF_FILENAME_MAX_LEN, CF_FILENAME_MAX_NAME, CF_FILENAME_MAX_PATH, CF_FindUnusedTransaction(), CF_NUM_TRANSACTONS_PER_PLAYBACK, CF_PERF_ID_DIRREAD, CF_Playback::cfdp_class, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_SUCCESS, CF_Engine::channels, CF_Playback::dest_id, CF_Playback::dir_id, CF_Playback::diopen, CF_TxnFilenames::dst_filename, CF_AppData_t::engine, os_dirent_t::FileName, CF_History::fnames, CF_Playback::fnames, CF_Transaction::history, CF_Playback::keep, CF_Playback::num_ts, OS_DirectoryClose(), OS_DirectoryRead(), CF_Transaction::pb, CF_Playback::priority, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_ProcessPlaybackDirectories() and CF_CFDP_ProcessPollingDirectories().

Here is the call graph for this function:



12.21.2.29 CF_CFDP_ProcessPollingDirectories() void CF_CFDP_ProcessPollingDirectories (CF_Channel_t * chan)

Kick the dir playback if timer elapsed.

Description

This function waits for the polling directory interval timer, and if it has expired, starts a playback in the polling directory.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>chan</i>	The channel associated with the playback
-------------	--

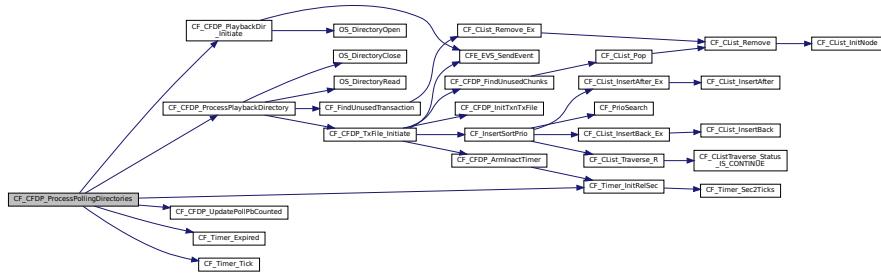
Definition at line 1480 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_UpdatePollPbCounted(), CF_MAX_POLLING_DIR_PER_CHAN, CF_Timer_Expired(), CF_Timer_InitRelSec(), CF_Timer_Tick(), CF_PollDir::cfdp_class, CF_ConfigTable::chan, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::config_table, CF_PollDir::dest_eid, CF_PollDir::dst_dir, CF_PollDir::enabled, CF_AppData_t::engine, CF_AppData_t::hk, CF_PollDir::interval_sec, CF_Poll::interval_timer, C

F_Playback::num_ts, CF_HkPacket::Payload, CF_Poll::pb, CF_Channel::poll, CF_HkChannel_Data::poll_counter, CF_ChannelConfig::polldir, CF_PollDir::priority, CF_PollDir::src_dir, and CF_Poll::timer_set.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.21.2.30 CF_CFDP_RecvAck() CFE_Status_t CF_CFDP_RecvAck (

```

    CFE_Transaction_t * txn,
    CFE_Logical_PduBuffer_t * ph
)
```

Unpack an ACK PDU from a received message.

This should only be invoked for buffers that have been identified as an acknowledgment PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

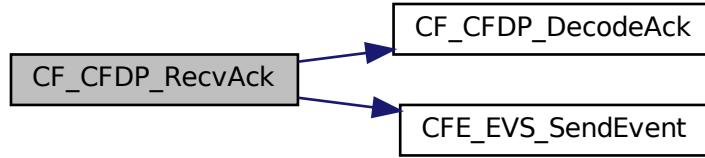
CFE_SUCCESS	on success
CF_SHORT_PDU_ERROR	on error

Definition at line 767 of file cf_cfdp.c.

References CF_Logical_IntHeader::ack, CF_CFDP_DecodeAck(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_ACK_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_R2_Recv_fin_ack(), and CF_CFDP_S2_WaitForEofAck().

Here is the call graph for this function:



12.21.2.31 CF_CFDP_RecvDrop() `void CF_CFDP_RecvDrop (`
 `CF_Transaction_t * txn,`
 `CF_Logical_PduBuffer_t * ph)`

Receive state function to ignore a packet.

Description

This function signature must match all receive state functions. The parameter `txn` is ignored here.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

Definition at line 836 of file cf_cfdp.c.

References `CF_AppData`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_HkRecv::dropped`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_HkCounters::recv`.

Referenced by `CF_CFDP_DispatchRecv()`.

12.21.2.32 CF_CFDP_RecvEof() `CFE_Status_t CF_CFDP_RecvEof (`
 `CF_Transaction_t * txn,`
 `CF_Logical_PduBuffer_t * ph)`

Unpack an EOF PDU from a received message.

This should only be invoked for buffers that have been identified as an end of file PDU.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

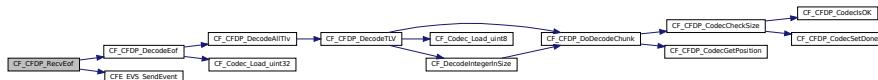
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 745 of file cf_cfdp.c.

References CF_CFDP_DecodeEof(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_EOF_SHORT_ERR←_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_Error, CFE_EVS_SendEvent(), CFE_SUCCESS, CF←Logical_IntHeader::eof, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_R_SubstateRecvEof().

Here is the call graph for this function:

**12.21.2.33 CF_CFDP_RecvFd()** *CFE_Status_t* CF_CFDP_RecvFd (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph
)
```

Unpack a file data PDU from a received message.

This should only be invoked for buffers that have been identified as a file data PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

<i>CFE_SUCCESS</i>	on success
--------------------	------------

Return values

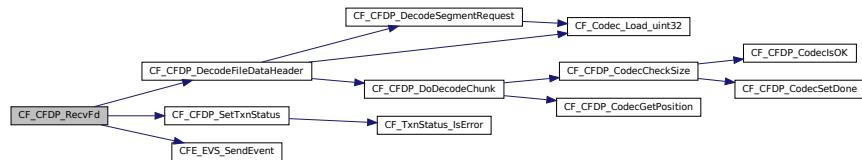
<i>CF_ERROR</i>	for general errors
<i>CF_SHORT_PDU_ERROR</i>	PDU too short

Definition at line 699 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_DecodeFileDataHeader(), CF_CFDP_SetTxnStatus(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_ERROR, CF_PDU_FD_SHORT_ERR_EID, CF_PDU_FD_UNSUPPOTED_ERR_EID, CF_SHORT_PDU_ERROR, CF_TxnStatus_PROTOCOL_ERROR, CFE_EVS_EventType_Error, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduHeader::crc_flag, CF_Logical_PduFileDataHeader::data_len, CF_HkRecv::error, CF_Logical_IntHeader::fd, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_Logical_PduBuffer::pdu_header, CF_HkCounters::recv, and CF_Logical_PduHeader::segment_meta_flag.

Referenced by CF_CFDP_R1_SubstateRecvFileData(), and CF_CFDP_R2_SubstateRecvFileData().

Here is the call graph for this function:



12.21.2.34 CF_CFDP_RecvFin() *CFE_Status_t* CF_CFDP_RecvFin (*CF_Transaction_t* * *txn*, *CF_Logical_PduBuffer_t* * *ph*)

Unpack an FIN PDU from a received message.

This should only be invoked for buffers that have been identified as a final PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

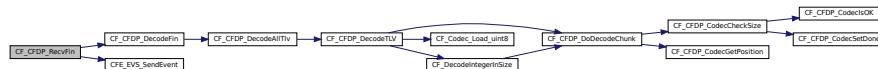
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 790 of file cf_cfdp.c.

References CF_CFDP_DecodeFin(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_FIN_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_IntHeader::fin, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



12.21.2.35 CF_CFDP_RecvIdle()

```
void CF_CFDP_RecvIdle (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Receive state function to process new rx transaction.

Description

An idle transaction has never had message processing performed on it. Typically, the first packet received for a transaction would be the metadata PDU. There's a special case for R2 where the metadata PDU could be missed, and filedatal comes in instead. In that case, an R2 transaction must still be started.

Assumptions, External Events, and Notes:

txn must not be NULL. There must be a received message.

Parameters

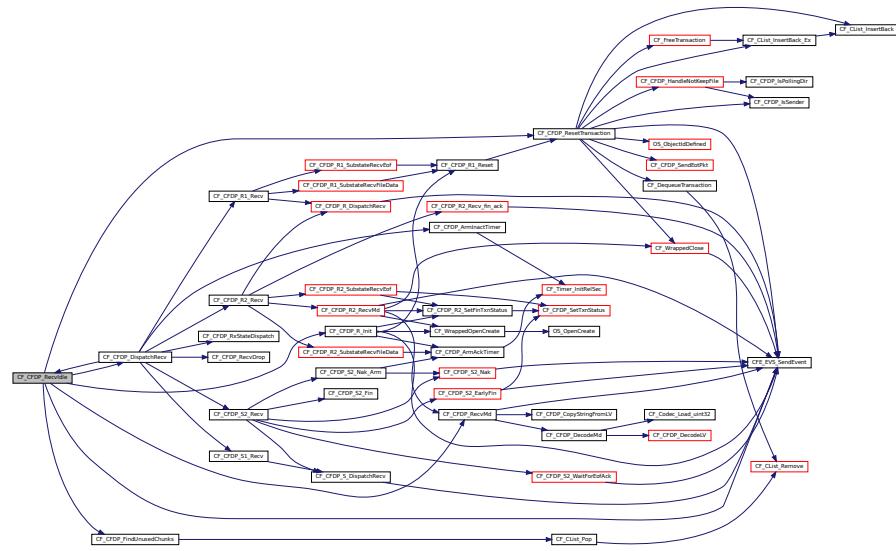
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 847 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_DispatchRecv(), CF_CFDP_FD_UNHANDLED_ERR_EID, CF_CFDP_FileDirective_METADATA, CF_CFDP_FindUnusedChunks(), CF_CFDP_IDLE_MD_ERR_EID, CF_CFDP_R_Init(), CF_CFDP_RecvMd(), CF_CFDP_ResetTransaction(), CF_Direction_RX, CF_TxnState_DROP, CF_TxnState_IDLE, CF_TxnState_R1, CF_TxnState_R2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::chunks, CF_HkChannel_Data::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_AppData_t::engine, CF_HkRecv::error, CF_Logical_PduBuffer::fdirective, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_History::peer_eid, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_History::src_eid, CF_Transaction::state, and CF_Logical_PduHeader::txm_mode.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



```
12.21.2.36 CF_CFDP_RecvMd() CFE_Status_t CF_CFDP_RecvMd (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

Unpack a metadata PDU from a received message.

This should only be invoked for buffers that have been identified as a metadata PDU.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

<i>CFE_SUCCESS</i>	on success
<i>CF_PDU_METADATA_ERROR</i>	on error

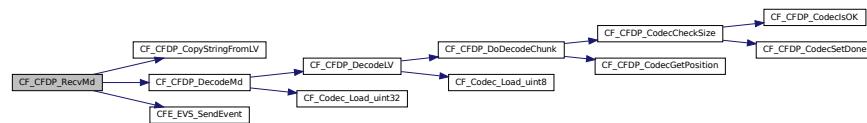
Definition at line 632 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CopyStringFromLV(), CF_CFDP_DecodeMd(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_INVALID_DST_LEN_ERR_EID, CF_PDU_INVALID_SRC_LEN_ERR_EID, CF_PDU_MD_RECVD_INF_EID, CF_PDU_MD_SHORT_ERR_EID, CF_PDU_METADATA_ERROR, CFE_EVS_EventType_EID

RROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduMd::dest_filename, CF_TxnFilenames::dst_filename, CF_HkRecv::error, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Logical_Lv::length, CF_Logical_IntHeader::md, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_HkCounters::recv, CF_Logical_PduMd::size, CF_Logical_PduMd::source_filename, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_R2_RecvMd(), and CF_CFDP_RecvIdle().

Here is the call graph for this function:



12.21.2.37 CF_CFDP_RecvNak() `CFE_Status_t` CF_CFDP_RecvNak (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Unpack a NAK PDU from a received message.

This should only be invoked for buffers that have been identified as a negative/non-acknowledgment PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

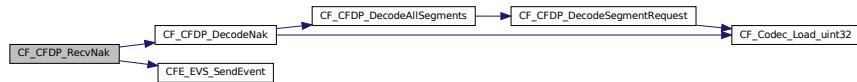
<code>CFE_SUCCESS</code>	on success
<code>CF_SHORT_PDU_ERROR</code>	on error

Definition at line 814 of file cf_cfdp.c.

References CF_CFDP_DecodeNak(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_NAK_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_PduBuffer::int_header, CF_Logical_IntHeader::nak, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_S2_Nak().

Here is the call graph for this function:



12.21.2.38 CF_CFDP_RecvPh() [CFE_Status_t](#) CF_CFDP_RecvPh (

```
    uint8 chan_num,
    CF_Logical_PduBuffer_t * ph )
```

Unpack a basic PDU header from a received message.

Description

This interprets the common PDU header and the file directive header (if applicable) and populates the logical PDU buffer.

Assumptions, External Events, and Notes:

A new message has been received.

Parameters

<i>chan_num</i>	The channel number for statistics purposes
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

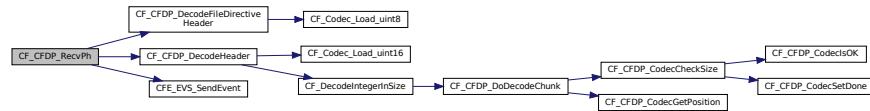
Return values

<i>CFE_SUCCESS</i>	on success
<i>CF_ERROR</i>	for general errors
<i>CF_SHORT_PDU_ERROR</i>	if PDU too short

Definition at line 572 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_DecodeFileDirectiveHeader(), CF_CFDP_DecodeHeader(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_ERROR, CF_NUM_CHANNELS, CF_PDU_LARGE_FILE_ERR_EID, CF_PDU_SHORT_HEADER_ERR_EID, CF_PDU_TRUNCATION_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Logical_PduBuffer::fdirective, CF_AppData_t::hk, CF_Logical_PduHeader::large_flag, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_HkRecv::pdu, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, and CF_HkCounters::recv.
Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



```
void CF_CFDP_ResetTransaction (
```

```
CF_Transaction_t * txn,  
bool keep_history )
```

Reset a transaction and all its internals to an unused state.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

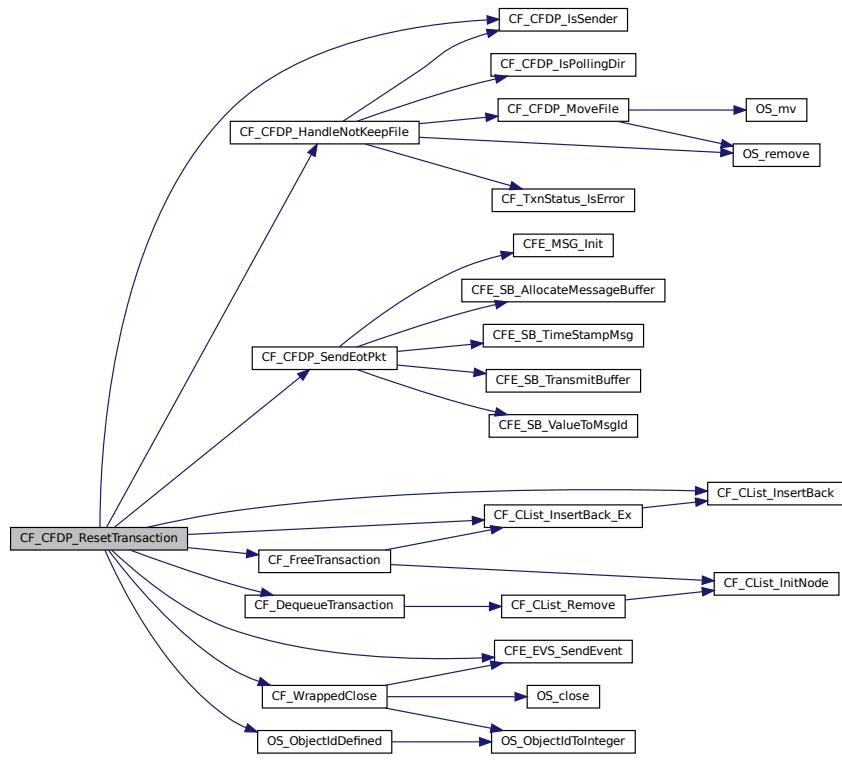
<i>txn</i>	Pointer to the transaction object
<i>keep_history</i>	Whether the transaction info should be preserved in history

Definition at line 1589 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_HandleNotKeepFile(), CF_CFDP_IsSender(), CF_CFDP_SendEotPkt(), CF_CList_InsertBack(), CF_CList_InsertBack_Ex(), CF_DequeueTransaction(), CF_Direction_TX, CF_FreeTransaction(), CF_NUM_CHANNELS, CF_QueueIdx_FREE, CF_QueueIdx_HIST, CF_QueueIdx_HIST_FREE, CF_RESET_FREED_XACT_DBG_EID, CF_WrappedClose(), CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_Engine::channels, CF_Transaction::chunks, CF_History::cl_node, CF_ChunkWrapper::cl_node, CF_Flags_Tx::cmd_tx, CF_StateFlags::com, CF_Channel::cs, CF_Channel::cur, CF_History::dir, CF_AppData_t::engine, CF_Transaction::fd, CF_Transaction::flags, CF_Transaction::history, CF_Transaction::keep, CF_Channel::num_cmd_tx, CF_Playback::num_ts, OS_ObjectIdDefined(), CF_Transaction::pb, CF_Flags_Common::q index, and CF_StateFlags::tx.

Referenced by CF_Abandon_TxnCmd(), CF_CFDP_R1_Reset(), CF_CFDP_RecvIdle(), CF_CFDP_S_Reset(), and CF_PurgeTransaction().

Here is the call graph for this function:



12.21.2.40 CF_CFDP_SendAck() `CFE_Status_t CF_CFDP_SendAck (`
`CF_Transaction_t * txn,`
`CF_CFDP_AckTxnStatus_t ts,`
`CF_CFDP_FileDirective_t dir_code,`
`CF_CFDP_ConditionCode_t cc,`
`CF_EntityId_t peer_eid,`
`CF_TransactionSeq_t tsn)`

Build an ACK PDU for transmit.

Assumptions, External Events, and Notes:

tx must not be NULL.

Note

`CF_CFDP_SendAck()` takes a `CF_TransactionSeq_t` instead of getting it from transaction history because of the special case where a FIN-ACK must be sent for an unknown transaction. It's better for long term maintenance to not build an incomplete `CF_History_t` for it.

Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Parameters

<i>ts</i>	Transaction ACK status
<i>dir_code</i>	File directive code being ACK'ed
<i>cc</i>	Condition code of transaction
<i>peer_eid</i>	Remote entity ID
<i>tsn</i>	Transaction sequence number

Returns

CFE_Status_t status code

Return values

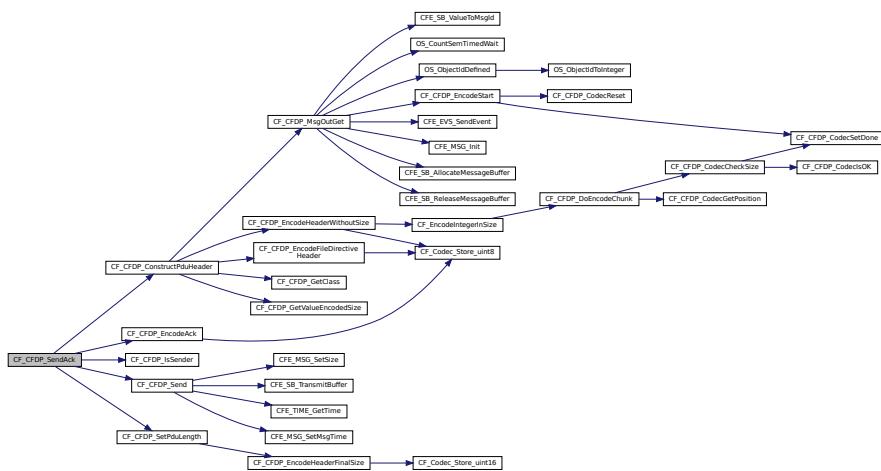
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 447 of file cf_cfdp.c.

References CF_Logical_IntHeader::ack, CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::cc, CF_AppData, CF_ASSERT, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeAck(), CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_EOF, CF_CFDP_FileDirective_FIN, CF_CFDP_IsSender(), CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_Logical_PduBuffer::penc, and CF_Logical_PduAck::txn_status.

Referenced by CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), and CF_CFDP_S_SubstateSendFinAck().

Here is the call graph for this function:



12.21.2.41 CF_CFDP_SendEof() `CFE_Status_t CF_CFDP_SendEof (`
`CF_Transaction_t * txn)`

Build an EOF PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

txn Pointer to the transaction object

Returns

CFE_Status_t status code

Return values

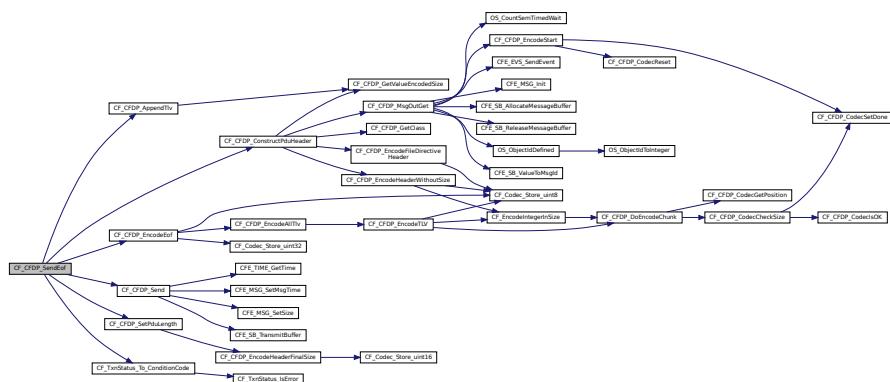
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 408 of file cf_cfdp.c.

References CF_Logical_PduEof::cc, CF_AppData, CF_CFDP_AppendTlv(), CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeEof(), CF_CFDP_FileDirective_EOF, CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_Txn_Status_To_ConditionCode(), CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Logical_PduEof::crc, CF_Transaction::crc, CF_Logical_ExtHeader::eof, CF_Transaction::fsize, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_Crc::result, CF_History::seq_num, CF_Logical_PduEof::size, CF_Logical_PduEof::tlv_list, and CF_History::txn_stat.

Referenced by CF CFDP S SendEof().

Here is the call graph for this function:



12.21.2.42 CF_CFDP_SendEotPkt() void CF_CFDP_SendEotPkt (

Send an end of transaction packet.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

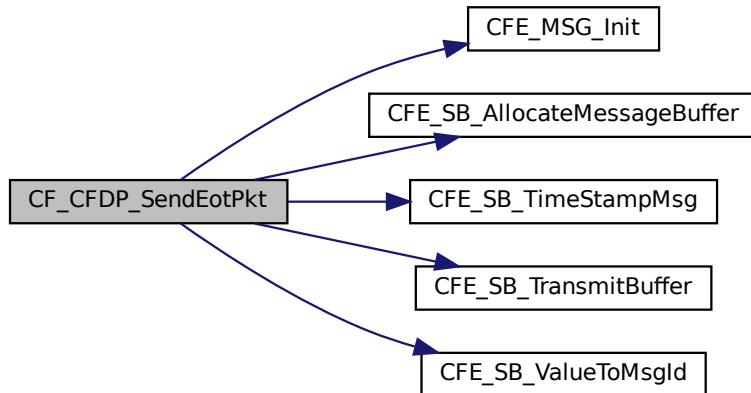
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 1673 of file cf_cfdp.c.

References CF_EOT_TLM_MID, CFE_MSG_Init(), CFE_SB_AllocateMessageBuffer(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitBuffer(), CFE_SB_ValueToMsgId(), CF_Transaction::chan_num, CF_EotPacket_Payload::channel, CF_Transaction::crc, CF_EotPacket_Payload::crc_result, CF_History::dir, CF_EotPacket_Payload::direction, CF_EotPacket_Payload::fnames, CF_History::fnames, CF_EotPacket_Payload::fsize, CF_Transaction::fsize, CF_Transaction::history, CF_EotPacket::Payload, CF_EotPacket_Payload::peer_eid, CF_History::peer_eid, CF_Crc::result, CF_EotPacket_Payload::seq_num, CF_History::seq_num, CF_EotPacket_Payload::src_eid, CF_History::src_eid, CF_EotPacket_Payload::state, CF_Transaction::state, CF_EotPacket::TelemetryHeader, CF_EotPacket_Payload::txn_stat, and CF_History::txn_stat.

Referenced by CF_CFDP_ResetTransaction().

Here is the call graph for this function:



12.21.2.43 CF_CFDP_SendFd() `CF_Status_t CF_CFDP_SendFd (`
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph)`

Send a previously-assembled filedata PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

Note

Unlike other "send" routines, the file data PDU must be acquired and filled by the caller prior to invoking this routine. This routine only sends the PDU that was previously allocated and assembled. As such, the typical failure possibilities do not apply to this call.

Returns

CFE_Status_t status code

Return values

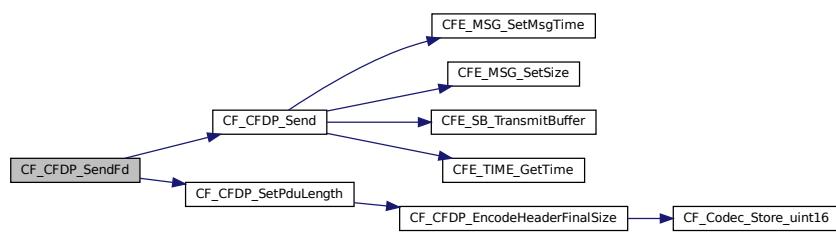
<code>CFE_SUCCESS</code>	on success. (error checks not yet implemented)
--------------------------	--

Definition at line 351 of file cf_cfdp.c.

References CF_CFDP_Send(), CF_CFDP_SetPduLength(), CFE_SUCCESS, and CF_Transaction::chan_num.

Referenced by CF_CFDP_S_SendFileData().

Here is the call graph for this function:



12.21.2.44 CF_CFDP_SendFin() `CFE_Status_t` CF_CFDP_SendFin (

 `CF_Transaction_t * txn,`

 `CF_CFDP_FinDeliveryCode_t dc,`

 `CF_CFDP_FinFileStatus_t fs,`

 `CF_CFDP_ConditionCode_t cc)`

Build a FIN PDU for transmit.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
<code>dc</code>	Final delivery status code (complete or incomplete)
<code>fs</code>	Final file status (retained or rejected, etc)
<code>cc</code>	Final CFDP condition code

Returns

CFE_Status_t status code

Return values

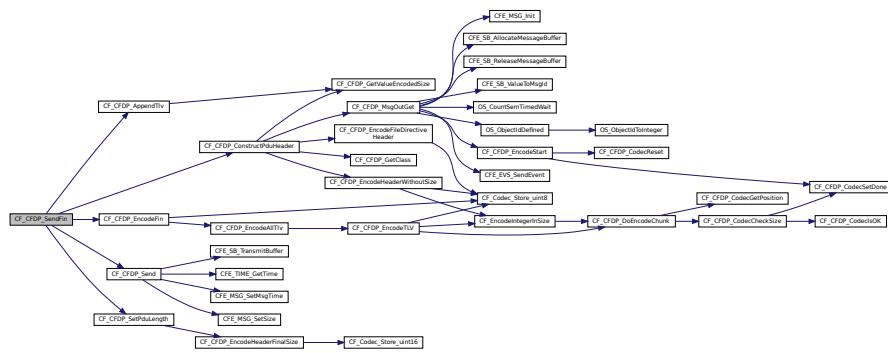
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 498 of file cf_cfdp.c.

References CF_Logical_PduFin::cc, CF_AppData, CF_CFDP_AppendTlv(), CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeFin(), CF_CFDP_FileDirective_FIN, CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Logical_PduFin::delivery_code, CF_Logical_PduFin::file_status, CF_Logical_ExtHeader::fin, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_History::seq_num, and CF_Logical_PduFin::tlv_list.

Referenced by CF_CFDP_R2_SubstateSendFin().

Here is the call graph for this function:



12.21.2.45 CF_CFDP_SendMd() CFE_Status_t CF_CFDP_SendMd (CF_Transaction_t * txn)

Build a metadata PDU for transmit.

Assumptions, External Events, and Notes

THERMOTROPIC NUBL

P. 100

txn Pointer to the transaction object

Returns

CEE_Status_t status_code

Return values

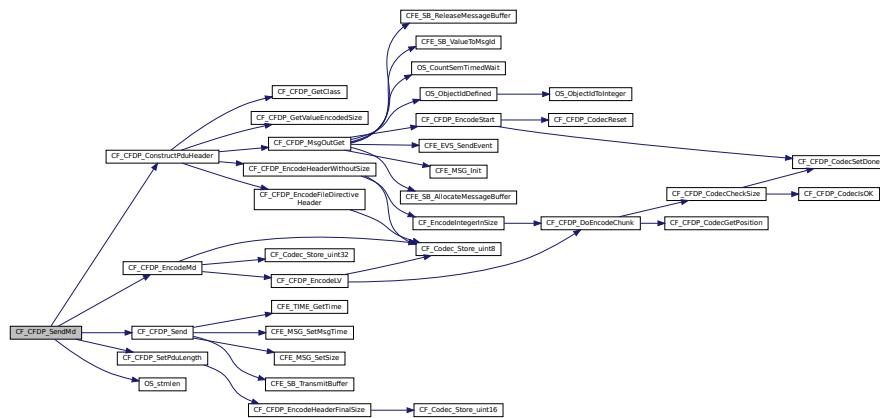
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 308 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeMd(), CF_CFDP_File↔Directive_METADATA, CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_TxnState_S1, CF_TxnState_S2, CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF↔_Logical_Lv::data_ptr, CF_Logical_PduMd::dest_filename, CF_TxnFilenames::dst_filename, CF_History::fnames, C↔F_Transaction::fsize, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_Logical_Lv::length, CF_Config↔Table::local_eid, CF_Logical_IntHeader::md, OS_strlen(), CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF↔_History::seq_num, CF_Logical_PduMd::size, CF_Logical_PduMd::source_filename, CF_TxnFilenames::src_filename, and CF_Transaction::state.

Referenced by CF_CFDP_S_CheckAndRespondNak(), and CF_CFDP_S_SubstateSendMetadata().

Here is the call graph for this function:



12.21.2.46 CF_CFDP_SendNak() CFE_Status_t CF_CFDP_SendNak (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Send a previously-assembled NAK PDU for transmit.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

Note

Unlike other "send" routines, the NAK PDU must be acquired and filled by the caller prior to invoking this routine. This routine only encodes and sends the previously-assembled PDU buffer. As such, the typical failure possibilities do not apply to this call.

Returns

CFE_Status_t status code

Return values

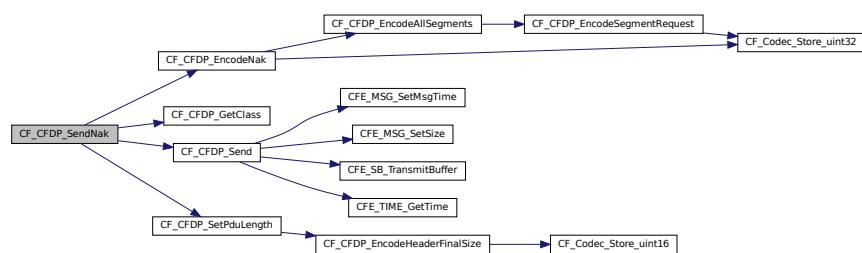
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 538 of file cf_cfdp.c.

References CF_ASSERT, CF_CFDP_CLASS_2, CF_CFDP_EncodeNak(), CF_CFDP_GetClass(), CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CFE_SUCCESS, CF_Transaction::chan←num, CF_Logical_PduBuffer::int_header, CF_Logical_IntHeader::nak, and CF_Logical_PduBuffer::penc.

Referenced by CF_CFDP_R_SubstateSendNak().

Here is the call graph for this function:

**12.21.2.47 CF_CFDP_SetPduLength()** static void CF_CFDP_SetPduLength (

CF_Logical_PduBuffer_t * ph) [static]

Definition at line 216 of file cf_cfdp.c.

References CF_CFDP_EncodeHeaderFinalSize(), CF_CODEC_GET_POSITION, CF_Logical_PduHeader::data←encoded_length, CF_Logical_PduHeader::header_encoded_length, CF_Logical_PduBuffer::pdu_header, and CF_Logical_PduBuffer::penc.

Referenced by CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFd(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), and CF_CFDP_SendNak().

Here is the call graph for this function:



```
12.21.2.48 CF_CFDP_SetTxnStatus() void CF_CFDP_SetTxnStatus (
    CF_Transaction_t * txn,
    CF_TxnStatus_t txn_stat )
```

Helper function to store transaction status code only.

This records the status in the history block but does not set FIN flag or take any other protocol/state machine actions.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>txn_stat</i>	Status Code value to set within transaction

Definition at line 1659 of file cf_cfdp.c.

References CF_TxnStatus_IsError(), CF_Transaction::history, and CF_History::txn_stat.

Referenced by CF_CFDP_CancelTransaction(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_Tick(), CF_CFDP_RecvFd(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_SubstateSendFileData(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().

Here is the call graph for this function:



```
12.21.2.49 CF_CFDP_TickTransactions() void CF_CFDP_TickTransactions (
```

```
    CF_Channel_t * chan )
```

Call R and then S tick functions for all active transactions.

Description

Traverses all transactions in the RX and TXW queues, and calls their tick functions. Note that the TXW queue is used twice: once for regular tick processing, and one for NAK response.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

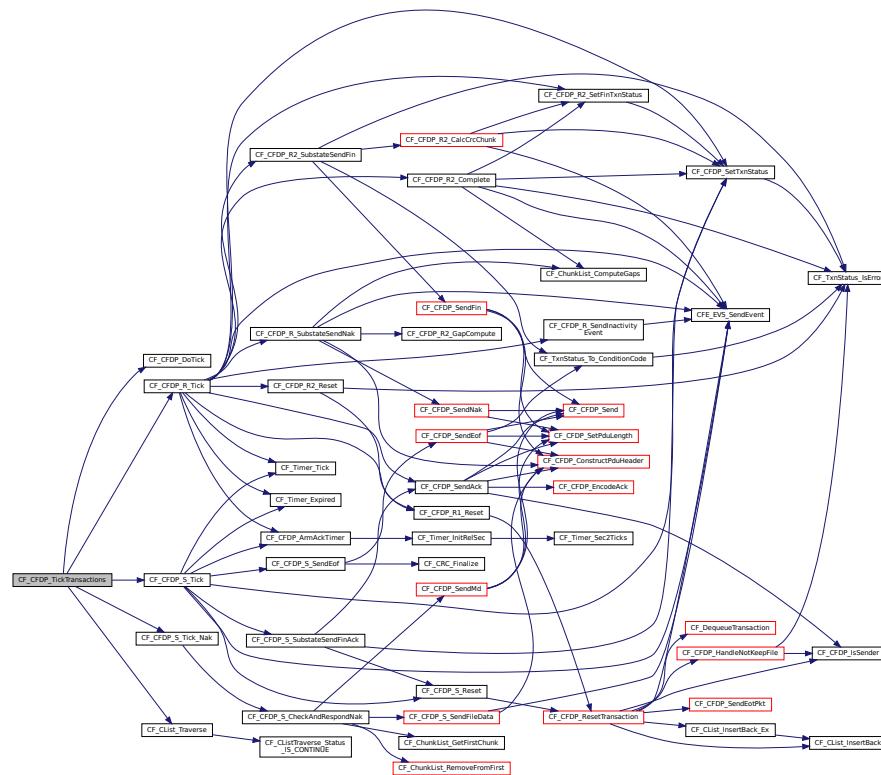
<i>chan</i>	Channel to tick
-------------	-----------------

Definition at line 1153 of file cf_cfdp.c.

References CF_ASSERT, CF_CFDP_DoTick(), CF_CFDP_R_Tick(), CF_CFDP_S_Tick(), CF_CFDP_S_Tick_Nak(), CF_CList_Traverse(), CF_QueueIdx_RX, CF_QueueIdx_TXW, CF_TickType_NUM_TYPES, CF_TickType_RX, CF_TickType_TXW_NAK, CF_CFDP_Tick_args::cont, CF_CFDP_Tick_args::early_exit, CF_Channel::qs, and CF_Channel::tick_type.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.21.2.50 CF_CFDP_TxFile() CFE_Status_t CF_CFDP_TxFile (

```
const char * src_filename,
const char * dst_filename,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
CF_EntityId_t dest_id )
```

Begin transmit of a file.

Description

This function sets up a transaction for and starts transmit of the given filename.

Assumptions, External Events, and Notes:

src filename must not be NULL. dst filename must not be NULL.

Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
-----------------------------	--

Returns

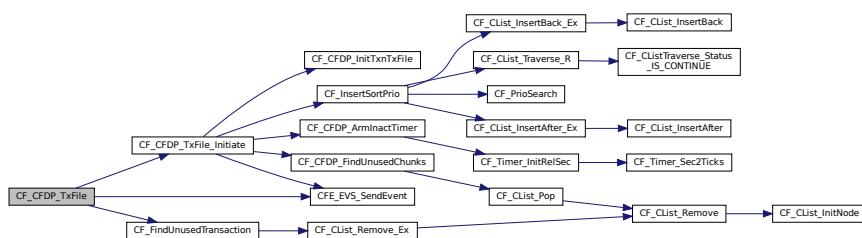
CFE_SUCCESS on success. CF_ERROR on error.

Definition at line 1262 of file cf_cfdp.c.

References CF_AppData, CF_ASSERT, CF_CFDP_MAX_CMD_TX_ERR_EID, CF_CFDP_TxFile_Initiate(), CF_EROR, CF_FindUnusedTransaction(), CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN, CF_NUM_CHAN, NELS, CF_TxnState_IDLE, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Engine::channels, CF_Flags_Tx::cmd_tx, CF_TxnFilenames::dst_filename, CF_AppData_t::engine, CF_Transaction::flags, CF_History::fnames, CF_Transaction::history, CF_Channel::num_cmd_tx, CF_TxnFilenames::src_filename, CF_Transaction::state, and CF_StateFlags::tx.

Referenced by CF_TxFileCmd().

Here is the call graph for this function:



```

12.21.2.51 CF_CFDP_TxFile_Initiate() static void CF_CFDP_TxFile_Initiate (
    CF_Transaction_t * txn,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority,
    CF_EntityId_t dest_id ) [static]
  
```

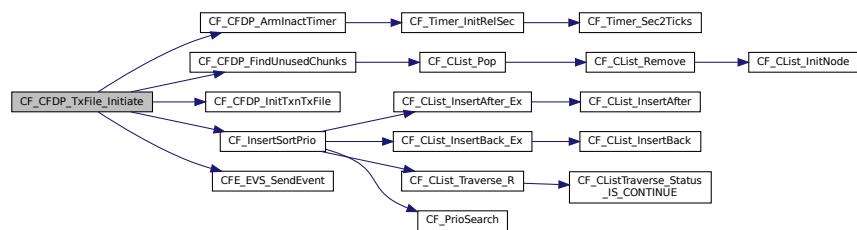
Definition at line 1229 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_ArmInactTimer(), CF_CFDP_FindUnusedChunks(), CF_CFDP_InitTxnTxFile(), CF_CFDP_S_START_SEND_INF_EID, CF_Direction_TX, CF_FILENAME_MAX_LEN, CF_InsertSortPrio(),

CF_Queueldx_PEND, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CF_Engine::channels, CF_Transaction::chunks, CF_AppData_t::config_table, CF_History::dir, CF_TxnFilenames::dst_filename, CF_AppData_t::engine, CF_History::fnames, CF_Transaction::history, CF_ConfigTable::local_eid, CF_History::peer_eid, CF_History::seq_num, CF_Engine::seq_num, CF_History::src_eid, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_ProcessPlaybackDirectory(), and CF_CFDP_TxFile().

Here is the call graph for this function:



12.21.2.52 CF_CFDP_UpdatePollPbCounted() static void CF_CFDP_UpdatePollPbCounted (
`CF_Playback_t * pb,`
`int up,`
`uint8 * counter) [static]`

Definition at line 1437 of file cf_cfdp.c.

References CF_ASSERT, and CF_Playback::counted.

Referenced by CF_CFDP_ProcessPlaybackDirectories(), and CF_CFDP_ProcessPollingDirectories().

12.22 apps/cf/fsw/src/cf_cfdp.h File Reference

```
#include "cf_cfdp_types.h"
```

Data Structures

- struct [CF_CFDP_CycleTx_args](#)
Structure for use with the [CF_CFDP_CycleTx\(\)](#) function.
- struct [CF_CFDP_Tick_args](#)
Structure for use with the [CF_CFDP_DoTick\(\)](#) function.

TypeDefs

- typedef struct [CF_CFDP_CycleTx_args](#) [CF_CFDP_CycleTx_args_t](#)
Structure for use with the [CF_CFDP_CycleTx\(\)](#) function.
- typedef struct [CF_CFDP_Tick_args](#) [CF_CFDP_Tick_args_t](#)
Structure for use with the [CF_CFDP_DoTick\(\)](#) function.

Functions

- void [CF_CFDP_EncodeStart](#) ([CF_EncoderState_t](#) *penc, void *msgbuf, [CF_Logical_PduBuffer_t](#) *ph, size_t encaps_hdr_size, size_t total_size)
Initiate the process of encoding a new PDU to send.

- void `CF_CFDP_DecodeStart` (`CF_DecoderState_t` *pdec, const void *msgbuf, `CF_Logical_PduBuffer_t` *ph, size_t encaps_hdr_size, size_t total_size)
Initiate the process of decoding a received PDU.
- void `CF_CFDP_ResetTransaction` (`CF_Transaction_t` *txn, bool keep_history)
Reset a transaction and all its internals to an unused state.
- void `CF_CFDP_SetTxnStatus` (`CF_Transaction_t` *txn, `CF_TxnStatus_t` txn_stat)
Helper function to store transaction status code only.
- void `CF_CFDP_SendEotPkt` (`CF_Transaction_t` *txn)
Send an end of transaction packet.
- `CFE_Status_t CF_CFDP_InitEngine` (void)
Initialization function for the CFDP engine.
- void `CF_CFDP_CycleEngine` (void)
Cycle the engine. Called once per wakeup.
- void `CF_CFDP_DisableEngine` (void)
Disables the CFDP engine and resets all state in it.
- `CFE_Status_t CF_CFDP_TxFile` (const char *src_filename, const char *dst_filename, `CF_CFDP_Class_t` cfcp_class, uint8 keep, uint8 chan, uint8 priority, `CF_EntityId_t` dest_id)
Begin transmit of a file.
- `CFE_Status_t CF_CFDP_PlaybackDir` (const char *src_filename, const char *dst_filename, `CF_CFDP_Class_t` cfcp_class, uint8 keep, uint8 chan, uint8 priority, uint16 dest_id)
Begin transmit of a directory.
- `CF_Logical_PduBuffer_t * CF_CFDP_ConstructPduHeader` (`const CF_Transaction_t` *txn, `CF_CFDP_FileDirective_t` directive_code, `CF_EntityId_t` src_eid, `CF_EntityId_t` dst_eid, bool towards_sender, `CF_TransactionSeq_t` tsn, bool silent)
Build the PDU header in the output buffer to prepare to send a packet.
- `CFE_Status_t CF_CFDP_SendMd` (`CF_Transaction_t` *txn)
Build a metadata PDU for transmit.
- `CFE_Status_t CF_CFDP_SendFd` (`CF_Transaction_t` *txn, `CF_Logical_PduBuffer_t` *ph)
Send a previously-assembled filedata PDU for transmit.
- `CFE_Status_t CF_CFDP_SendEof` (`CF_Transaction_t` *txn)
Build an EOF PDU for transmit.
- `CFE_Status_t CF_CFDP_SendAck` (`CF_Transaction_t` *txn, `CF_CFDP_AckTxnStatus_t` ts, `CF_CFDP_FileDirective_t` dir_code, `CF_CFDP_ConditionCode_t` cc, `CF_EntityId_t` peer_eid, `CF_TransactionSeq_t` tsn)
Build an ACK PDU for transmit.
- `CFE_Status_t CF_CFDP_SendFin` (`CF_Transaction_t` *txn, `CF_CFDP_FinDeliveryCode_t` dc, `CF_CFDP_FinFileStatus_t` fs, `CF_CFDP_ConditionCode_t` cc)
Build a FIN PDU for transmit.
- `CFE_Status_t CF_CFDP_SendNak` (`CF_Transaction_t` *txn, `CF_Logical_PduBuffer_t` *ph)
Send a previously-assembled NAK PDU for transmit.
- void `CF_CFDP_AppendTlv` (`CF_Logical_TlvList_t` *ptlv_list, `CF_CFDP_TlvType_t` tlv_type)
Appends a single TLV value to the logical PDU data.
- `CFE_Status_t CF_CFDP_RecvPh` (uint8 chan_num, `CF_Logical_PduBuffer_t` *ph)
Unpack a basic PDU header from a received message.
- `CFE_Status_t CF_CFDP_RecvMd` (`CF_Transaction_t` *txn, `CF_Logical_PduBuffer_t` *ph)
Unpack a metadata PDU from a received message.
- `CFE_Status_t CF_CFDP_RecvFd` (`CF_Transaction_t` *txn, `CF_Logical_PduBuffer_t` *ph)
Unpack a file data PDU from a received message.
- `CFE_Status_t CF_CFDP_RecvEof` (`CF_Transaction_t` *txn, `CF_Logical_PduBuffer_t` *ph)

- **`CFE_Status_t CF_CFDP_RecvAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Unpack an EOF PDU from a received message.
- **`CFE_Status_t CF_CFDP_RecvFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Unpack an ACK PDU from a received message.
- **`CFE_Status_t CF_CFDP_RecvNak (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Unpack an FIN PDU from a received message.
- **`void CF_CFDP_DispatchRecv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Dispatch received packet to its handler.
- **`void CF_CFDP_CancelTransaction (CF_Transaction_t *txn)`**

Cancels a transaction.
- **`void CF_CFDP_InitTxnTxFile (CF_Transaction_t *txn, CF_CFDP_Class_t cfdp_class, uint8 keep, uint8 chan, uint8 priority)`**

Helper function to set tx file state in a transaction.
- **`int CF_CFDP_CopyStringFromLV (char *buf, size_t buf_maxsz, const CF_Logical_Lv_t *src_lv)`**

Copy string data from a lv (length, value) pair.
- **`void CF_CFDP_ArmAckTimer (CF_Transaction_t *txn)`**

Arm the ACK timer.
- **`void CF_CFDP_RecvDrop (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Receive state function to ignore a packet.
- **`void CF_CFDP_RecvIdle (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

Receive state function to process new rx transaction.
- **`CF_CListTraverse_Status_t CF_CFDP_CloseFiles (CF_CListNode_t *node, void *context)`**

List traversal function to close all files in all active transactions.
- **`void CF_CFDP_CycleTx (CF_Channel_t *chan)`**

Cycle the current active tx or make a new one active.
- **`CF_CListTraverse_Status_t CF_CFDP_CycleTxFirstActive (CF_CListNode_t *node, void *context)`**

List traversal function that cycles the first active tx.
- **`void CF_CFDP_TickTransactions (CF_Channel_t *chan)`**

Call R and then S tick functions for all active transactions.
- **`void CF_CFDP_ProcessPlaybackDirectory (CF_Channel_t *chan, CF_Playback_t *pb)`**

Step each active playback directory.
- **`void CF_CFDP_ProcessPollingDirectories (CF_Channel_t *chan)`**

Kick the dir playback if timer elapsed.
- **`CF_CListTraverse_Status_t CF_CFDP_DoTick (CF_CListNode_t *node, void *context)`**

List traversal function that calls a r or s tick function.
- **`bool CF_CFDP_IsPollingDir (const char *src_file, uint8 chan_num)`**

Check if source file came from polling directory.
- **`void CF_CFDP_HandleNotKeepFile (CF_Transaction_t *txn)`**

Remove/Move file after transaction.
- **`void CF_CFDP_MoveFile (const char *src, const char *dest_dir)`**

Move File.

12.22.1 Detailed Description

The CF Application CFDP engine and packet parsing header file

12.22.2 Typedef Documentation

12.22.2.1 CF_CFDP_CycleTx_args_t `typedef struct CF_CFDP_CycleTx_args CF_CFDP_CycleTx_args_t`
Structure for use with the [CF_CFDP_CycleTx\(\)](#) function.

12.22.2.2 CF_CFDP_Tick_args_t `typedef struct CF_CFDP_Tick_args CF_CFDP_Tick_args_t`
Structure for use with the [CF_CFDP_DoTick\(\)](#) function.

12.22.3 Function Documentation

12.22.3.1 CF_CFDP_AppendTlv() `void CF_CFDP_AppendTlv (`
`CF_Logical_TlvList_t * ptlv_list,`
`CF_CFDP_TlvType_t tlv_type)`

Appends a single TLV value to the logical PDU data.

This function implements common functionality between SendEof and SendFin which append a TLV value specifying the faulting entity ID.

Assumptions, External Events, and Notes:

`ptlv_list` must not be NULL. Only `CF_CFDP_TLV_TYPE_ENTITY_ID` type is currently implemented

Parameters

<code>ptlv_list</code>	TLV list from current PDU buffer.
<code>tlv_type</code>	Type of TLV to append. Currently must be <code>CF_CFDP_TLV_TYPE_ENTITY_ID</code> .

Definition at line 371 of file cf_cfdp.c.

References `CF_AppData`, `CF_CFDP_GetValueEncodedSize()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_PDU_MAX_TLV`, `CF_AppData_t::config_table`, `CF_Logical_Tlv::data`, `CF_Logical_TlvData::data_ptr`, `CF_Logical_TlvData::eid`, `CF_Logical_Tlv::length`, `CF_ConfigTable::local_eid`, `CF_Logical_TlvList::num_tlv`, `CF_Logical_TlvList::tlv`, and `CF_Logical_Tlv::type`.

Referenced by `CF_CFDP_SendEof()`, and `CF_CFDP_SendFin()`.

Here is the call graph for this function:



12.22.3.2 CF_CFDP_ArmAckTimer() `void CF_CFDP_ArmAckTimer (`
`CF_Transaction_t * txn)`

Arm the ACK timer.

Description

Helper function to arm the ACK timer and set the flag.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

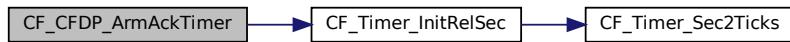
<i>txn</i>	Pointer to the transaction state
------------	----------------------------------

Definition at line 123 of file cf_cfdp.c.

References CF_Transaction::ack_timer, CF_Flags_Common::ack_timer_armed, CF_ChannelConfig::ack_timer_s, C← F_AppData, CF_Timer_InitRelSec(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_StateFlags::com, CF← AppData_t::config_table, and CF_Transaction::flags.

Referenced by CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_Init(), CF_CFDP_R_Tick(), CF_CFDP_S2← Nak_Arm(), and CF_CFDP_S_Tick().

Here is the call graph for this function:



12.22.3.3 CF_CFDP_CancelTransaction()

```
void CF_CFDP_CancelTransaction (
    CF_Transaction_t * txn )
```

Cancels a transaction.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

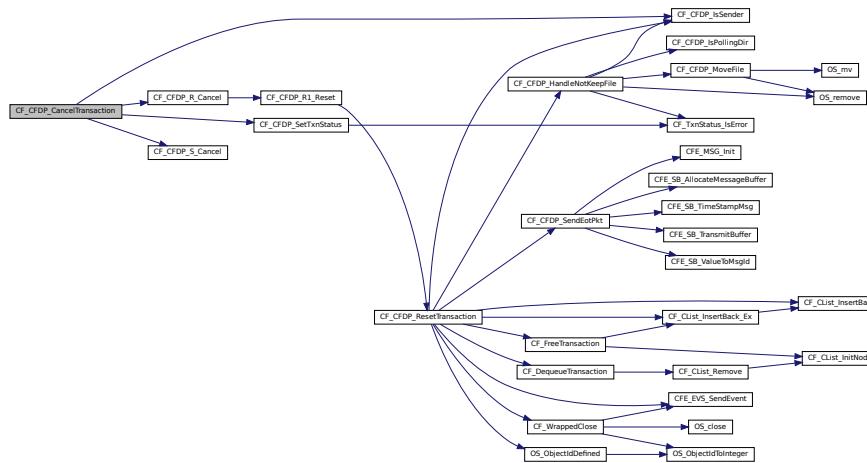
<i>txn</i>	Pointer to the transaction state
------------	----------------------------------

Definition at line 1734 of file cf_cfdp.c.

References CF_Flags_Common::canceled, CF_CFDP_IsSender(), CF_CFDP_R_Cancel(), CF_CFDP_S_Cancel(), CF_CFDP_SetTxnStatus(), CF_TxnStatus_CANCEL_REQUEST_RECEIVED, CF_StateFlags::com, and CF← Transaction::flags.

Referenced by CF_Cancel_TxnCmd().

Here is the call graph for this function:



12.22.3.4 CF_CFDP_CloseFiles() `CF_CListTraverse_Status_t CF_CFDP_CloseFiles (`
`CF_CListNode_t * node,`
`void * context)`

List traversal function to close all files in all active transactions.

This helper is used in conjunction with [CF_CList_Traverse\(\)](#).

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<code>node</code>	List node pointer
<code>context</code>	Opaque pointer, not used in this function

Returns

integer traversal code

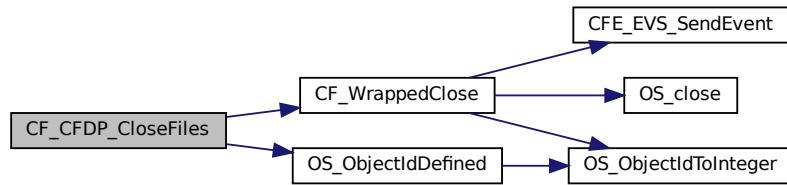
Return values

<code>Always</code>	CF_LIST_CONT indicate list traversal should not exit early.
---------------------	---

Definition at line 1751 of file cf_cfdp.c.

References CF_CLIST_CONT, CF_WrappedClose(), container_of, CF_Transaction::fd, and OS_ObjectIdDefined().
Referenced by CF_CFDP_DisableEngine().

Here is the call graph for this function:



```

12.22.3.5 CF_CFDP_ConstructPduHeader() CF_Logical_PduBuffer_t* CF_CFDP_ConstructPduHeader (
    const CF_Transaction_t * txn,
    CF_CFDP_FileDirective_t directive_code,
    CF_EntityId_t src_eid,
    CF_EntityId_t dst_eid,
    bool towards_sender,
    CF_TransactionSeq_t tsn,
    bool silent )
  
```

Build the PDU header in the output buffer to prepare to send a packet.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>directive_code</i>	Code to use for file directive headers (set to 0 for data)
<i>src_eid</i>	Value to set in source entity ID field
<i>dst_eid</i>	Value to set in destination entity ID field
<i>towards_sender</i>	Whether this is transmitting toward the sender entity
<i>tsn</i>	Transaction sequence number to put into PDU
<i>silent</i>	If true, suppress error event if no message buffer available

Returns

Pointer to PDU buffer which may be filled with additional data

Return values

<i>NULL</i>	if no message buffer available
-------------	--------------------------------

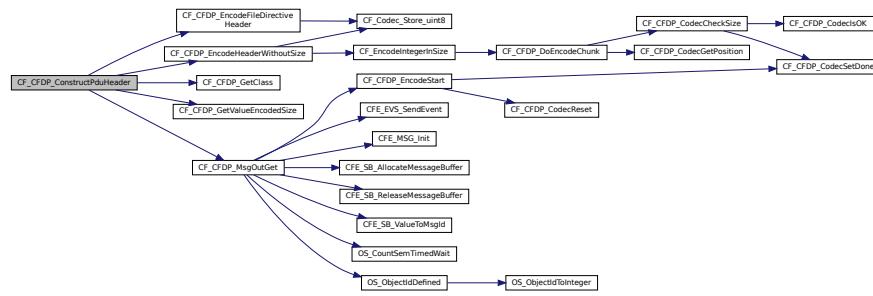
Definition at line 238 of file cf_cfdp.c.

References CF_CFDP_CLASS_1, CF_CFDP_EncodeFileDirectiveHeader(), CF_CFDP_EncodeHeaderWithoutSize(), CF_CFDP_GetClass(), CF_CFDP_GetValueEncodedSize(), CF_CFDP_MsgOutGet(), CF_Logical_Pdu

Header::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduFileDirectiveHeader::directive_code, CF_Logical_PduHeader::eid_length, CF_Logical_PduBuffer::fdirective, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_Logical_PduBuffer::penc, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_R_SubstateSendNak(), CF_CFDP_S_SendFileData(), CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFin(), and CF_CFDP_SendMd().

Here is the call graph for this function:



12.22.3.6 CF_CFDP_CopyStringFromLV()

```
int CF_CFDP_CopyStringFromLV (
    char * buf,
    size_t buf_maxsz,
    const CF_Logical_Lv_t * src_lv )
```

Copy string data from a lv (length, value) pair.

This copies a string value from an LV pair inside a PDU buffer. In CF this is used for file names embedded within PDUs.

Note

This function assures that the output string is terminated appropriately, such that it can be used as a normal C string. As such, the buffer size must be at least 1 byte larger than the maximum string length.

Assumptions, External Events, and Notes:

src_lv must not be NULL. buf must not be NULL.

Parameters

<i>buf</i>	Pointer to buffer to store string
<i>buf_maxsz</i>	Total size of buffer pointer to by buf (usable size is 1 byte less, for termination)
<i>src_lv</i>	Pointer to LV pair from logical PDU buffer

Returns

The resulting string length, NOT including termination character

Return values

CF_ERROR	on error
----------	----------

Definition at line 1714 of file cf_cfdp.c.

References CF_ERROR, CF_Logical_Lv::data_ptr, and CF_Logical_Lv::length.

Referenced by CF_CFDP_RecvMd().

12.22.3.7 CF_CFDP_CycleEngine()

```
void CF_CFDP_CycleEngine (
```

```
    void )
```

Cycle the engine. Called once per wakeup.

Assumptions, External Events, and Notes:

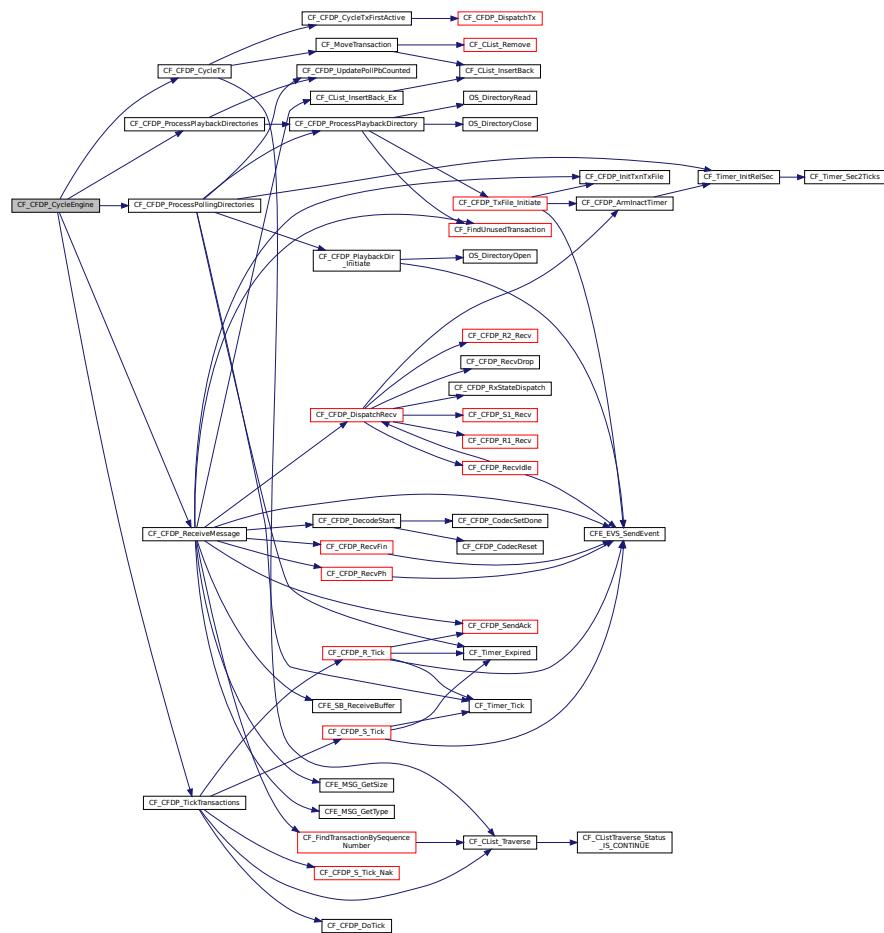
None

Definition at line 1549 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CycleTx(), CF_CFDP_ProcessPlaybackDirectories(), CF_CFDP_ProcessPollingDirectories(), CF_CFDP_ReceiveMessage(), CF_CFDP_TickTransactions(), CF_NUM_CHANNELS, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Engine::enabled, CF_AppData_t::engine, CF_HkChannel_Data::frozen, CF_AppData_t::hk, CF_Engine::outgoing_counter, and CF_HkPacket::Payload.

Referenced by CF_WakeupCmd().

Here is the call graph for this function:



12.22.3.8 CF_CFDP_CycleTx() void CF_CFDP_CycleTx (CF_Channel_t * chan)

Cycle the current active tx or make a new one active.

Description

First traverses all tx transactions on the active queue. If at least one is found, then it stops. Otherwise it moves a transaction on the pending queue to the active queue and tries again to find an active one.

Assumptions, External Events, and Notes:

None

Parameters

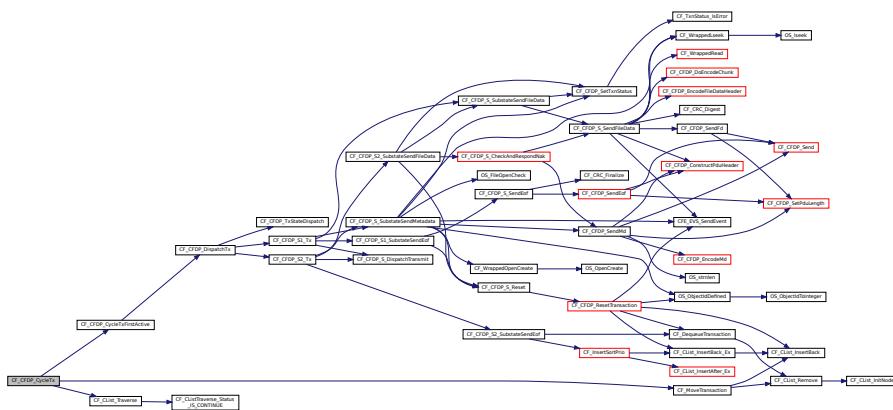
chan Channel to cycle

Definition at line 1077 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CycleTxFirstActive(), CF_CList_Traverse(), CF_MoveTransaction(), CF_QueueIdx_PEND, CF_QueueIdx_TXA, CF_ConfigTable::chan, CF_Engine::channels, CF_AppData_t::config_table, container_of, CF_Channel::cur, CF_ChannelConfig::dequeue_enabled, CF_AppData_t::engine, CF_Channel::qs, and CF_CFDP_CycleTx_args::ran_one.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



```
12.22.3.9 CF_CFDP_CycleTxFirstActive() CF_CListTraverse_Status_t CF_CFDP_CycleTxFirstActive (
```

List traversal function that cycles the first active tx.

This helper is used in conjunction with `CE_CList_Traverse()`

Description

There can only be one active tx transaction per engine cycle. This function finds the first active, and then sends file data PDUs until there are no outgoing message buffers.

Assumptions, External Events, and Notes:

node must not be NULL. Context must not be NULL.

Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_CycleTx_args_t object (passed through)

Returns

integer traversal code

Return values

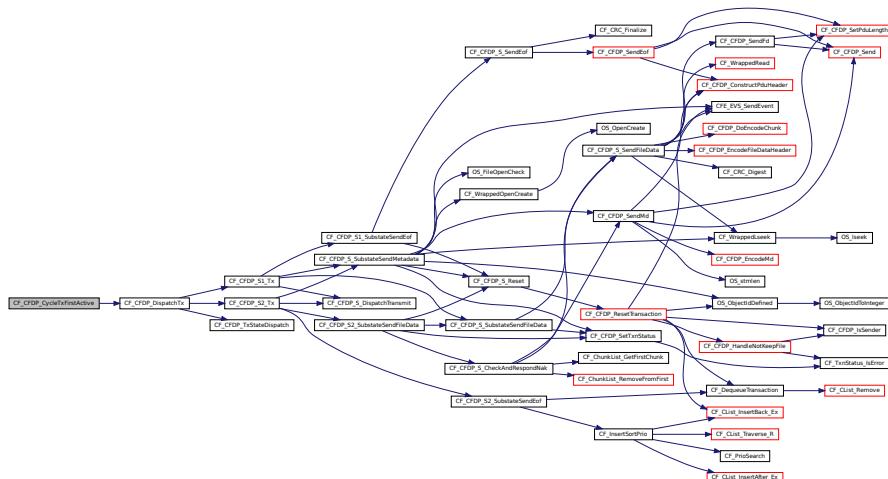
<code>CF_CLIST_EXIT</code>	when it's found, which terminates list traversal
<code>CF_CLIST_CONT</code>	when it's isn't found, which causes list traversal to continue

Definition at line 1041 of file cf_cfdp.c.

References CF_ASSERT, CF_CFDP_DispatchTx(), CF_CLIST_CONT, CF_CLIST_EXIT, CF_PERF_ID_PDUSENT, CF_QueueIdx_TXA, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CF_CFDP_CycleTx_args::chan, CF_Transaction::chan_num, CF_StateFlags::com, container_of, CF_Channel::cur, CF_Transaction::flags, CF_Flags_Common::q_index, CF_CFDP_CycleTx_args::ran_one, and CF_Flags_Common::suspended.

Referenced by CF_CFDP_CycleTx().

Here is the call graph for this function:



12.22.3.10 CF_CFDP_DecodeStart() void CF_CFDP_DecodeStart (

```
CF_DecoderState_t * pdec,  
const void * msgbuf,  
CF_Logical_PduBuffer_t * ph,  
size_t encaps_hdr_size,  
size_t total_size )
```

Initiate the process of decoding a received PDU.

This resets the decoder and PDU buffer to initial values, and prepares for decoding a new PDU that was received from a remote entity.

Parameters

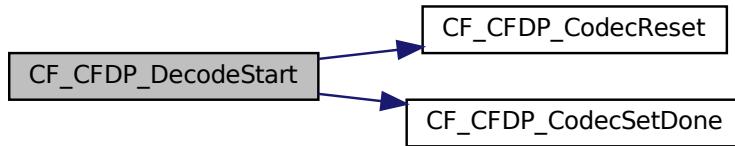
<i>pdec</i>	Decoder state structure, will be reset-initialized by this call to point to msgbuf.
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Total size of msgbuf encapsulation structure (decoding cannot exceed this)

Definition at line 89 of file cf_cfdp.c.

References CF_DecoderState::base, CF_CFDP_CodecReset(), CF_CFDP_CodecSetDone(), CF_DecoderState::codec_state, CF_CodecState::max_size, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



12.22.3.11 CF_CFDP_DisableEngine() void CF_CFDP_DisableEngine (void)

Disables the CFDP engine and resets all state in it.

Assumptions, External Events, and Notes:

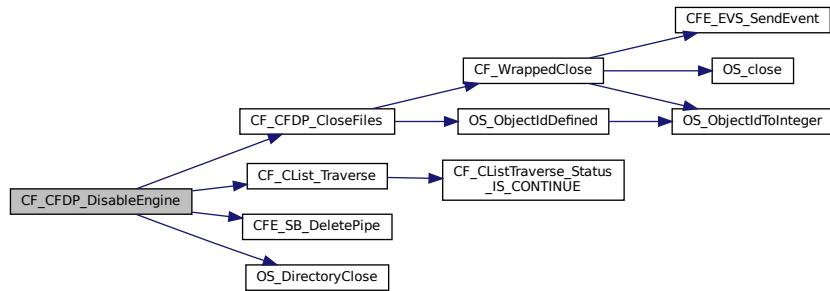
None

Definition at line 1767 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_CloseFiles(), CF_CList_Traverse(), CF_MAX_COMMAND_ED_PLAYBACK_DIRECTORIES_PER_CHAN, CF_MAX_POLLING_DIR_PER_CHAN, CF_NUM_CHANNELS, CF_QueueIdx_RX, CF_QueueIdx_TXA, CF_QueueIdx_TXW, CFE_SB_DeletePipe(), CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Playback::dir_id, CF_Engine::enabled, CF_AppData_t::engine, CF_AppData_t::hk, OS_DirectoryClose(), CF_HkPacket::Payload, CF_Poll::pb, CF_Channel::pipe, CF_Channel::playback, CF_Channel::poll, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_DisableEngineCmd().

Here is the call graph for this function:



12.22.3.12 CF_CFDP_DispatchRecv() `void CF_CFDP_DispatchRecv (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)`

Dispatch received packet to its handler.

This dispatches the PDU to the appropriate handler based on the transaction state

Assumptions, External Events, and Notes:

`txn` must not be null. It must be an initialized transaction.

Parameters

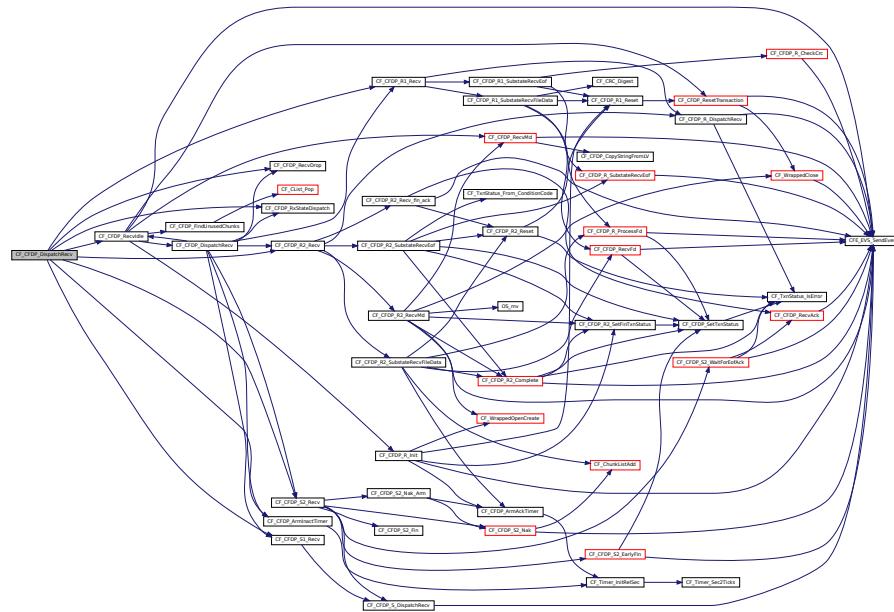
<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

Definition at line 169 of file cf_cfdp.c.

References CF_CFDP_ArmInactTimer(), CF_CFDP_R1_Recv(), CF_CFDP_R2_Recv(), CF_CFDP_RecvDrop(), CF_CFDP_RecvIdle(), CF_CFDP_RxStateDispatch(), CF_CFDP_S1_Recv(), CF_CFDP_S2_Recv(), CF_TxnState_DR_OP, CF_TxnState_IDLE, CF_TxnState_R1, CF_TxnState_R2, CF_TxnState_S1, CF_TxnState_S2, and CF_CFDP_TxnRecvDispatchTable_t::rx.

Referenced by CF_CFDP_ReceiveMessage(), and CF_CFDP_RecvIdle().

Here is the call graph for this function:



```
12.22.3.13 CF_CFDP_DoTick() CF_CListTraverse_Status_t CF_CFDP_DoTick (
```

<pre> CF_CListNode_t * node,</pre>	<pre> void * context)</pre>
---------------------------------------	---------------------------------

List traversal function that calls a r or s tick function.
This helper is used in conjunction with [CF_CList_Traverse\(\)](#).

Assumptions, External Events, and Notes:

node must not be NULL, context must not be NULL.

Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_Tick_args_t object (passed through)

Returns

integer traversal code

Return values

<code>CF_CLIST_EXIT</code>	when it's found, which terminates list traversal
<code>CF_CLIST_CONT</code>	when it's isn't found, which causes list traversal to continue

Definition at line 1120 of file cf_cfdp.c.

References CF CLIST CONT_CF CLIST EXIT_CF CF_CFDP Tick args::chan, CF StateFlags::com, CF_CFDP Tick<-->

_args::cont, container_of, CF_Channel::cur, CF_CFDP_Tick_args::early_exit, CF_Transaction::flags, CF_CFDP_Tick_args::fn, and CF_Flags_Common::suspended.
Referenced by CF_CFDP_TickTransactions().

12.22.3.14 CF_CFDP_EncodeStart() void CF_CFDP_EncodeStart (

```
CF_EncoderState_t * penc,
void * msgbuf,
CF_Logical_PduBuffer_t * ph,
size_t encaps_hdr_size,
size_t total_size )
```

Initiate the process of encoding a new PDU to send.

This resets the encoder and PDU buffer to initial values, and prepares for encoding a new PDU for sending to a remote entity.

Parameters

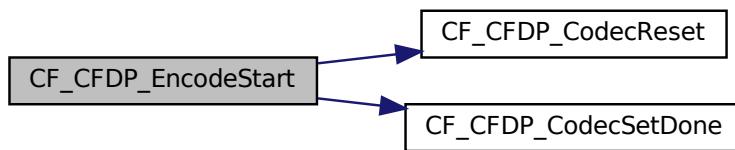
<i>penc</i>	Encoder state structure, will be reset-initialized by this call to point to msgbuf.
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Allocated size of msgbuf encapsulation structure (encoding cannot exceed this)

Definition at line 55 of file cf_cfdp.c.

References CF_EncoderState::base, CF_CFDP_CodecReset(), CF_CFDP_CodecSetDone(), CF_EncoderState::codec_state, CF_CodecState::max_size, and CF_Logical_PduBuffer::penc.

Referenced by CF_CFDP_MsgOutGet().

Here is the call graph for this function:



12.22.3.15 CF_CFDP_HandleNotKeepFile() void CF_CFDP_HandleNotKeepFile (

```
CF_Transaction_t * txn )
```

Remove/Move file after transaction.

This helper is used to handle "not keep" file option after a transaction.

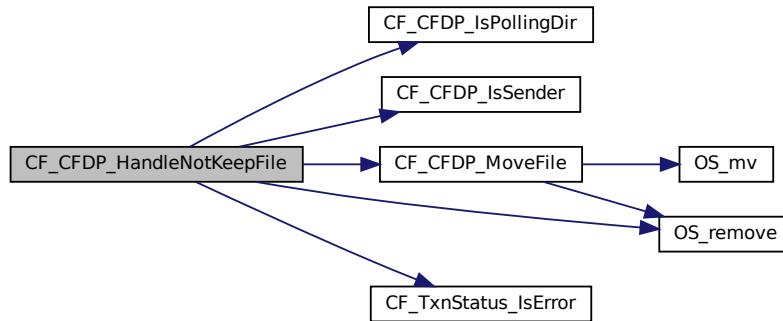
Assumptions, External Events, and Notes:

Definition at line 1850 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_IsPollingDir(), CF_CFDP_IsSender(), CF_CFDP_MoveFile(), CF_TxnStatus_IsError(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_TxnFilenames::dst_filename, CF_ConfigTable::fail_dir, CF_History::fnames, CF_Transaction::history, CF_ChannelConfig::move_dir, OS_remove(), CF_TxnFilenames::src_filename, and CF_History::txn_stat.

Referenced by CF_CFDP_ResetTransaction().

Here is the call graph for this function:



12.22.3.16 CF_CFDP_InitEngine() `CFE_Status_t CF_CFDP_InitEngine (void)`

Initialization function for the CFDP engine.

Description

Performs all initialization of the CFDP engine

Assumptions, External Events, and Notes:

Only called once.

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
--------------------------	--

Returns

anything else on error.

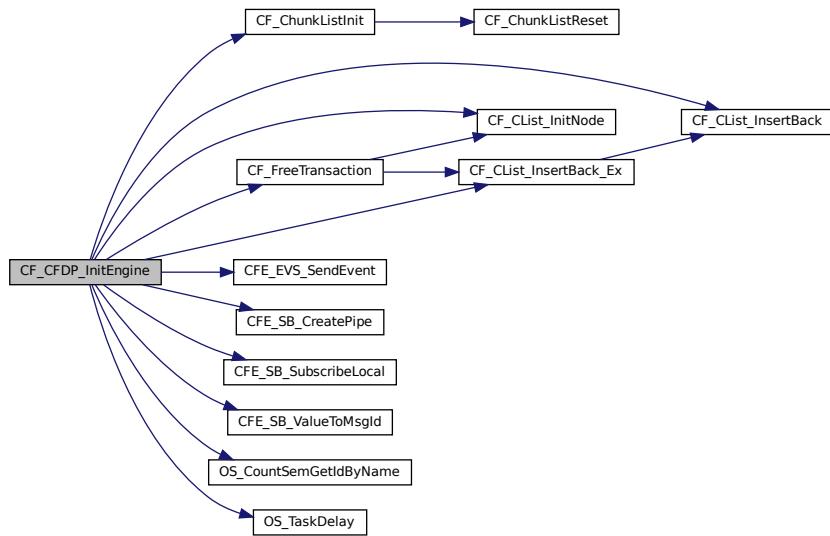
Definition at line 931 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION, CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION, CF_CHANNEL_PIPE_PREFIX, CF_ChunkListInit(), CF_CList_InitNode(), CF_CList_InsertBack(), CF_CList_InsertBack_Ex(), CF_CR_CHANNEL_PIPE_ERR_EID, CF_Direction_NUM,

CF_FreeTransaction(), CF_INIT_SEM_ERR_EID, CF_INIT_SUB_ERR_EID, CF_NUM_CHANNELS, CF_NUM_CHANNELS_ALL_CHANNELS, CF_NUM_HISTORIES_PER_CHANNEL, CF_NUM_TRANSACTIONS_PER_CHANNEL, CF_QueueIdx_HIST_FREE, CF_STARTUP_SEM_MAX_RETRIES, CF_STARTUP_SEM_TASK_DELAY, CFE_EVT_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_CreatePipe(), CFE_SB_SubscribeLocal(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CF_ConfigTable::chan, CF_Transaction::chan_num, CF_Engine::channels, CF_Engine::chunk_mem, CF_ChunkWrapper::chunks, CF_Engine::chunks, CF_History::cl_node, CF_ChunkWrapper::cl_node, CF_AppData_t::config_table, CF_Channel::cs, CF_Engine::enabled, CF_AppData_t::engine, CF_Engine::histories, CF_ChannelConfig::mid_input, OS_CountSemGetIdByName(), OS_ERR_NAME_NOT_FOUND, OS_SUCCESS, OS_TaskDelay(), CF_Channel::pipe, CF_ChannelConfig::pipe_depth_input, CF_Channel::sem_id, CF_ChannelConfig::sem_name, and CF_Engine::transactions.

Referenced by CF_AppInit(), and CF_EnableEngineCmd().

Here is the call graph for this function:



12.22.3.17 CF_CFDP_InitTxnTxFile() void CF_CFDP_InitTxnTxFile (

```

    CF_Transaction_t * txn,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority
  )
  
```

Helper function to set tx file state in a transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for sending a file.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>cfdp_class</i>	Set to class 1 or class 2

Parameters

<i>keep</i>	Whether to keep the local file
<i>chan</i>	CF channel number
<i>priority</i>	Priority of transfer

Definition at line 1216 of file cf_cfdp.c.

References CF_TxnState_S1, CF_TxnState_S2, CF_Transaction::chan_num, CF_Transaction::keep, CF_Transaction::priority, and CF_Transaction::state.

Referenced by CF_CFDP_ReceiveMessage(), and CF_CFDP_TxFile_Initiate().

```
12.22.3.18 CF_CFDP_IsPollingDir() bool CF_CFDP_IsPollingDir (
    const char * src_file,
    uint8 chan_num )
```

Check if source file came from polling directory.

Assumptions, External Events, and Notes:

Return values

<i>true/false</i>	
-------------------	--

Definition at line 1816 of file cf_cfdp.c.

References CF_AppData, CF_FILENAME_MAX_LEN, CF_MAX_POLLING_DIR_PER_CHAN, CF_ConfigTable::chan, CF_AppData_t::config_table, CF_ChannelConfig::polldir, and CF_PollDir::src_dir.

Referenced by CF_CFDP_HandleNotKeepFile().

```
12.22.3.19 CF_CFDP_MoveFile() void CF_CFDP_MoveFile (
    const char * src,
    const char * dest_dir )
```

Move File.

This helper is used to move a file.

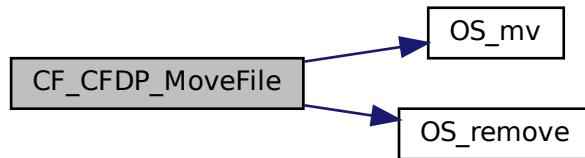
Assumptions, External Events, and Notes:

Definition at line 1884 of file cf_cfdp.c.

References OS_ERROR, OS_MAX_PATH_LEN, OS_mv(), OS_remove(), and OS_SUCCESS.

Referenced by CF_CFDP_HandleNotKeepFile().

Here is the call graph for this function:



12.22.3.20 CF_CFDP_PlaybackDir() CFE_Status_t CF_CFDP_PlaybackDir (

```

const char * src_filename,
const char * dst_filename,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
uint16 dest_id )
  
```

Begin transmit of a directory.

Description

This function sets up CF_Playback_t structure with state so it can become part of the directory polling done at each engine cycle.

Assumptions, External Events, and Notes:

src_filename must not be NULL. *dst_filename* must not be NULL.

Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
-----------------------------	--

Returns

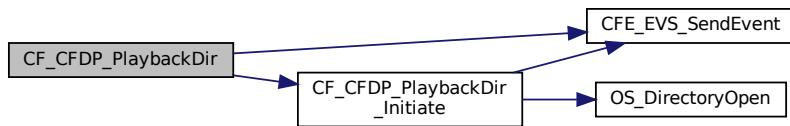
CFE_SUCCESS on success. CF_ERROR on error.

Definition at line 1343 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_DIR_SLOT_ERR_EID, CF_CFDP_PlaybackDir_Initiate(), CF_ERROR, CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN, CFE_ES_EventType_ERROR, CFE_EVS_SendEvent(), CF_Engine::channels, CF_AppData_t::engine, and CF_Channel::playback.

Referenced by CF_PlaybackDirCmd().

Here is the call graph for this function:



12.22.3.21 CF_CFDP_ProcessPlaybackDirectory() void CF_CFDP_ProcessPlaybackDirectory (
 CF_Channel_t * chan,
 CF_Playback_t * pb)

Step each active playback directory.

Description

Check if a playback directory needs iterated, and if so does, and if a valid file is found initiates playback on it.

Assumptions, External Events, and Notes:

chan must not be NULL, pb must not be NULL.

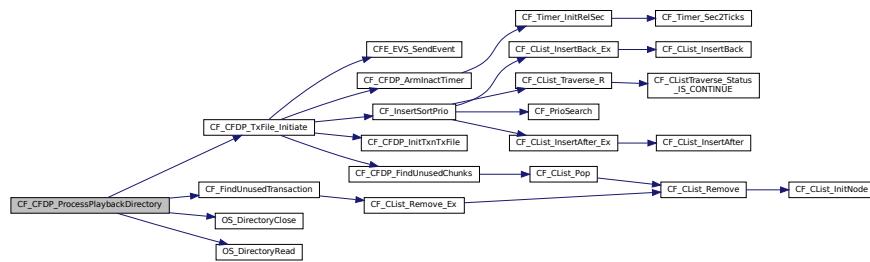
Parameters

<i>chan</i>	The channel associated with the playback
<i>pb</i>	The playback state

Definition at line 1373 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_ASSERT, CF_CFDP_TxFile_Initiate(), CF_FILENAME_MAX_LEN, CF_FILENAME_MAX_NAME, CF_FILENAME_MAX_PATH, CF_FindUnusedTransaction(), CF_NUM_TRANSACTONS_PER_PLAYBACK, CF_PERF_ID_DIRREAD, CF_Playback::cfdp_class, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_SUCCESS, CF_Engine::channels, CF_Playback::dest_id, CF_Playback::dir_id, CF_Playback::diopen, CF_TxnFilenames::dst_filename, CF_AppData_t::engine, os_dirent_t::FileName, CF_History::fnames, CF_Playback::fnames, CF_Transaction::history, CF_Playback::keep, CF_Playback::num_ts, OS_DirectoryClose(), OS_DirectoryRead(), CF_Transaction::pb, CF_Playback::priority, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_ProcessPlaybackDirectories(), and CF_CFDP_ProcessPollingDirectories().
Here is the call graph for this function:



12.22.3.22 CF_CFDP_ProcessPollingDirectories()

```
void CF_CFDP_ProcessPollingDirectories (
    CF_Channel_t * chan )
```

Kick the dir playback if timer elapsed.

Description

This function waits for the polling directory interval timer, and if it has expired, starts a playback in the polling directory.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

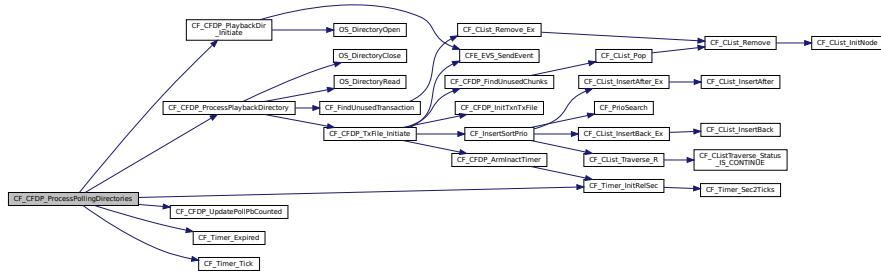
<code>chan</code>	The channel associated with the playback
-------------------	--

Definition at line 1480 of file cf_cfdp.c.

References CF_Playback::busy, CF_AppData, CF_CFDP_PlaybackDir_Initiate(), CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_UpdatePollPbCounted(), CF_MAX_POLLING_DIR_PER_CHAN, CF_Timer_Expired(), C_F_Timer_InitRelSec(), CF_Timer_Tick(), CF_PollDir::cfdp_class, CF_ConfigTable::chan, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::config_table, CF_PollDir::dest_eid, CF_PollDir::dst_dir, CF_PollDir::enabled, CF_AppData_t::engine, CF_AppData_t::hk, CF_PollDir::interval_sec, CF_Poll::interval_timer, C_F_Playback::num_ts, CF_HkPacket::Payload, CF_Poll::pb, CF_Channel::poll, CF_HkChannel_Data::poll_counter, CF_ChannelConfig::polldir, CF_PollDir::priority, CF_PollDir::src_dir, and CF_Poll::timer_set.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.22.3.23 CF_CFDP_RecvAck() *CFE_Status_t* CF_CFDP_RecvAck (

```

    CFE_Transaction_t * txn,
    CFE_Logical_PduBuffer_t * ph
  )
```

Unpack an ACK PDU from a received message.

This should only be invoked for buffers that have been identified as an acknowledgment PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

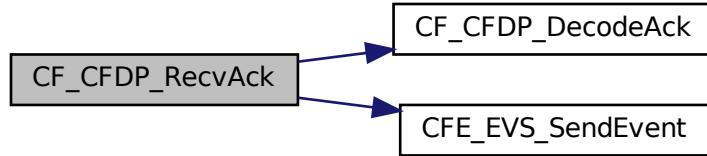
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 767 of file cf_cfdp.c.

References CF_Logical_IntHeader::ack, CF_CFDP_DecodeAck(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_ACK_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_R2_Recv_fin_ack(), and CF_CFDP_S2_WaitForEofAck().

Here is the call graph for this function:



12.22.3.24 CF_CFDP_RecvDrop() `void CF_CFDP_RecvDrop (`
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph)`

Receive state function to ignore a packet.

Description

This function signature must match all receive state functions. The parameter `txn` is ignored here.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

Definition at line 836 of file cf_cfdp.c.

References CF_AppData, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkRecv::dropped, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_HkCounters::recv.

Referenced by CF_CFDP_DispatchRecv().

12.22.3.25 CF_CFDP_RecvEof() `CFE_Status_t CF_CFDP_RecvEof (`
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph)`

Unpack an EOF PDU from a received message.

This should only be invoked for buffers that have been identified as an end of file PDU.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

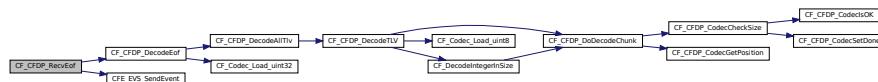
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 745 of file cf_cfdp.c.

References CF_CFDP_DecodeEof(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_EOF_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_Error, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_ItHeader::eof, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_R_SubstateRecvEof().

Here is the call graph for this function:

**12.22.3.26 CF_CFDP_RecvFd()** *CFE_Status_t* CF_CFDP_RecvFd (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph
)
```

Unpack a file data PDU from a received message.

This should only be invoked for buffers that have been identified as a file data PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

<i>CFE_SUCCESS</i>	on success
--------------------	------------

Return values

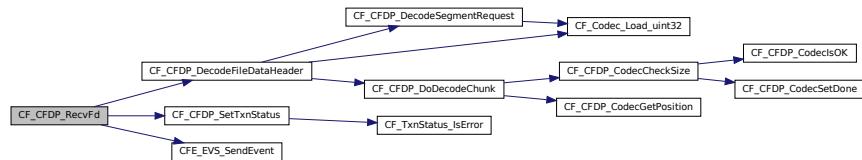
<i>CF_ERROR</i>	for general errors
<i>CF_SHORT_PDU_ERROR</i>	PDU too short

Definition at line 699 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_DecodeFileDataHeader(), CF_CFDP_SetTxnStatus(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_ERROR, CF_PDU_FD_SHORT_ERR_EID, CF_PDU_FD_UNSUPPOTED_ERR_EID, CF_SHORT_PDU_ERROR, CF_TxnStatus_PROTOCOL_ERROR, CFE_EVS_EventType_Error, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduHeader::crc_flag, CF_Logical_PduFileDataHeader::data_len, CF_HkRecv::error, CF_Logical_IntHeader::fd, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_Logical_PduBuffer::pdu_header, CF_HkCounters::recv, and CF_Logical_PduHeader::segment_meta_flag.

Referenced by CF_CFDP_R1_SubstateRecvFileData(), and CF_CFDP_R2_SubstateRecvFileData().

Here is the call graph for this function:



12.22.3.27 CF_CFDP_RecvFin() *CFE_Status_t* CF_CFDP_RecvFin (*CF_Transaction_t* * *txn*, *CF_Logical_PduBuffer_t* * *ph*)

Unpack an FIN PDU from a received message.

This should only be invoked for buffers that have been identified as a final PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

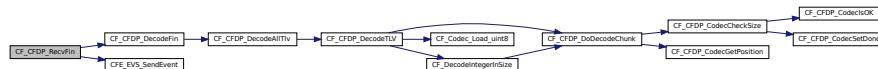
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 790 of file cf_cfdp.c.

References CF_CFDP_DecodeFin(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_FIN_SHORT_ERR_←
EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF←
Logical_IntHeader::fin, CF_Logical_PduBuffer::int_header, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



12.22.3.28 CF_CFDP_RecvIdle()

```
void CF_CFDP_RecvIdle (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Receive state function to process new rx transaction.

Description

An idle transaction has never had message processing performed on it. Typically, the first packet received for a transaction would be the metadata PDU. There's a special case for R2 where the metadata PDU could be missed, and filedatal comes in instead. In that case, an R2 transaction must still be started.

Assumptions, External Events, and Notes:

txn must not be NULL. There must be a received message.

Parameters

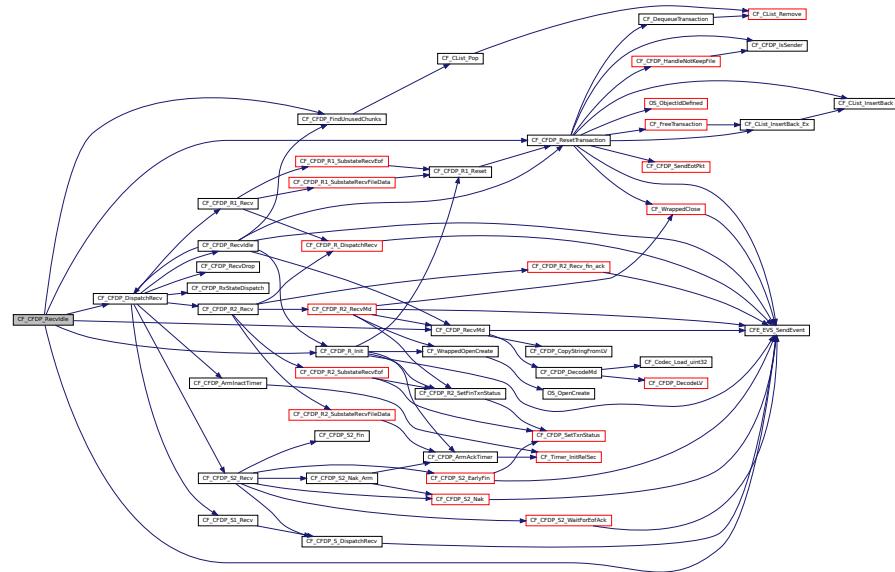
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 847 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_DispatchRecv(), CF_CFDP_FD_UNHANDLED_ERR_EID, CF_CFDP_File←
Directive_METADATA, CF_CFDP_FindUnusedChunks(), CF_CFDP_IDLE_MD_ERR_EID, CF_CFDP_R_Init(), CF←
_CFDP_RecvMd(), CF_CFDP_ResetTransaction(), CF_Direction_RX, CF_TxnState_DROP, CF_TxnState_IDLE, C←
F_TxnState_R1, CF_TxnState_R2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan←
_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::chunks, CF_HkChannel_Data←
::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_AppData_t::engine, CF_HkRecv::error, CF←
Logical_PduBuffer::fdirective, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md←
_recv, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_History←
::peer_eid, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_Logical_PduHeader::sequence_num,
CF_Logical_PduHeader::source_eid, CF_History::src_eid, CF_Transaction::state, and CF_Logical_PduHeader::txm←
_mode.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.22.3.29 CF_CFDP_RecvMd() *CFE_Status_t* CF_CFDP_RecvMd (
 CF_Transaction_t * *txn*,
 CF_Logical_PduBuffer_t * *ph*)

Unpack a metadata PDU from a received message.
This should only be invoked for buffers that have been identified as a metadata PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

<i>CFE_SUCCESS</i>	on success
<i>CF_PDU_METADATA_ERROR</i>	on error

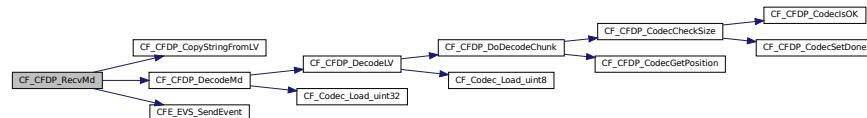
Definition at line 632 of file cf_cfdp.c.

References CF_AppData, CF_CFDP_CopyStringFromLV(), CF_CFDP_DecodeMd(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_INVALID_DST_LEN_ERR_EID, CF_PDU_INVALID_SRC_LEN_ERR_EID, CF_PDU_M

D_RECVD_INF_EID, CF_PDU_MD_SHORT_ERR_EID, CF_PDU_METADATA_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduMd::dest_filename, CF_TxnFilenames::dst_filename, CF_HkRecv::error, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Logical_Lv::length, CF_Logical_IntHeader::md, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_HkCounters::recv, CF_Logical_PduMd::size, CF_Logical_PduMd::source_filename, and CF_TxnFilenames::src_filename.

Referenced by CF_CFDP_R2_RecvMd(), and CF_CFDP_RecvIdle().

Here is the call graph for this function:



12.22.3.30 CF_CFDP_RecvNak() `CFE_Status_t` CF_CFDP_RecvNak (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Unpack a NAK PDU from a received message.

This should only be invoked for buffers that have been identified as a negative/non-acknowledgment PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

Returns

integer status code

Return values

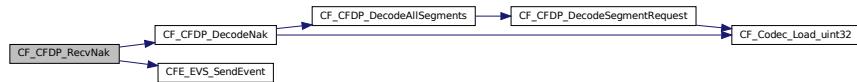
<code>CFE_SUCCESS</code>	on success
<code>CF_SHORT_PDU_ERROR</code>	on error

Definition at line 814 of file cf_cfdp.c.

References CF_CFDP_DecodeNak(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_PDU_NAK_SHORT_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Logical_PduBuffer::int_header, CF_Logical_IntHeader::nak, and CF_Logical_PduBuffer::pdec.

Referenced by CF_CFDP_S2_Nak().

Here is the call graph for this function:



12.22.3.31 CF_CFDP_RecvPh()

```
CFE_Status_t CF_CFDP_RecvPh (
    uint8 chan_num,
    CF_Logical_PduBuffer_t * ph )
```

Unpack a basic PDU header from a received message.

Description

This interprets the common PDU header and the file directive header (if applicable) and populates the logical PDU buffer.

Assumptions, External Events, and Notes:

A new message has been received.

Parameters

<i>chan_num</i>	The channel number for statistics purposes
<i>ph</i>	The logical PDU buffer being received

Returns

integer status code

Return values

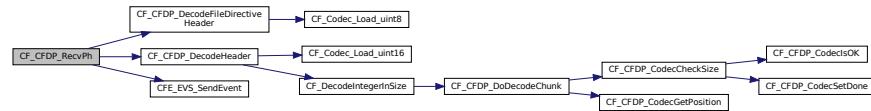
<i>CFE_SUCCESS</i>	on success
<i>CF_ERROR</i>	for general errors
<i>CF_SHORT_PDU_ERROR</i>	if PDU too short

Definition at line 572 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_DecodeFileDirectiveHeader(), CF_CFDP_DecodeHeader(), CF_CODEC_GET_SIZE, CF_CODEC_IS_OK, CF_ERROR, CF_NUM_CHANNELS, CF_PDU_LARGE_FILE_ERR_EID, CF_PDU_SHORT_HEADER_ERR_EID, CF_PDU_TRUNCATION_ERR_EID, CF_SHORT_PDU_ERROR, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Logical_PduBuffer::fdirective, CF_AppData_t::hk, CF_Logical_PduHeader::large_flag, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdec, CF_HkRecv::pdu, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, and CF_HkCounters::recv.

Referenced by CF_CFDP_ReceiveMessage().

Here is the call graph for this function:



12.22.3.32 CF_CFDP_ResetTransaction() void CF_CFDP_ResetTransaction (
`CF_Transaction_t * txn,`
`bool keep_history)`

Reset a transaction and all its internals to an unused state.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

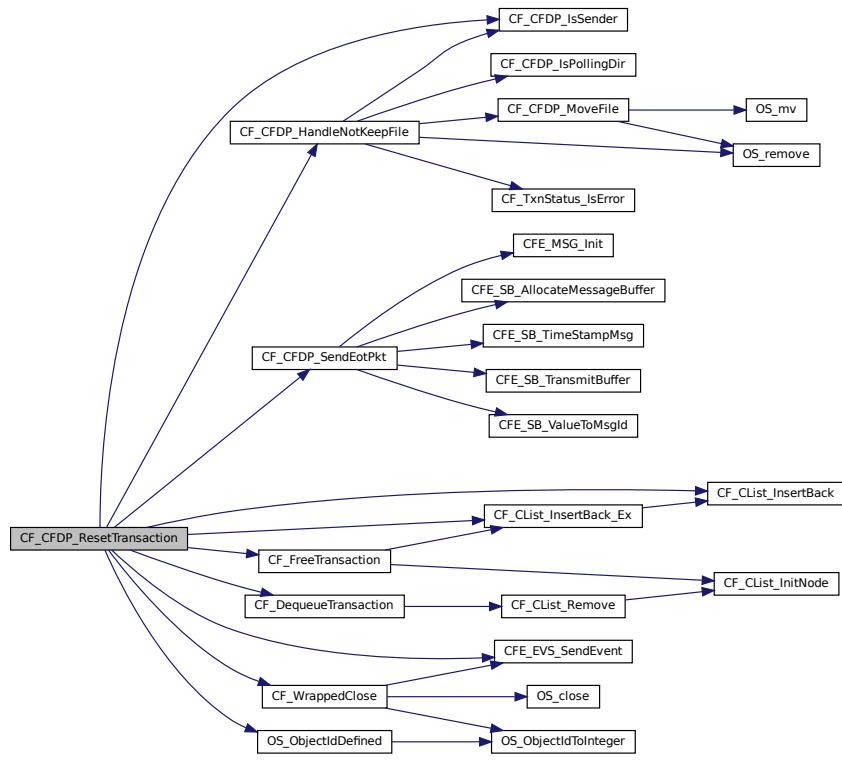
<i>txn</i>	Pointer to the transaction object
<i>keep_history</i>	Whether the transaction info should be preserved in history

Definition at line 1589 of file cf_cfdp.c.

References CF_AppData, CF_ASSERT, CF_CFDP_HandleNotKeepFile(), CF_CFDP_IsSender(), CF_CFDP_SendEotPkt(), CF_CList_InsertBack(), CF_CList_InsertBack_Ex(), CF_DequeueTransaction(), CF_Direction_TX, CF_FreeTransaction(), CF_NUM_CHANNELS, CF_QueueIdx_FREE, CF_QueueIdx_HIST, CF_QueueIdx_HIST_FREE, CF_RESET_FREED_XACT_DBG_EID, CF_WrappedClose(), CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_Engine::channels, CF_Transaction::chunks, CF_History::cl_node, CF_ChunkWrapper::cl_node, CF_Flags_Tx::cmd_tx, CF_StateFlags::com, CF_Channel::cs, CF_Channel::cur, CF_History::dir, CF_AppData_t::engine, CF_Transaction::fd, CF_Transaction::flags, CF_Transaction::history, CF_Transaction::keep, CF_Channel::num_cmd_tx, CF_Playback::num_ts, OS_ObjectIdDefined(), CF_Transaction::pb, CF_Flags_Common::q_index, and CF_StateFlags::tx.

Referenced by CF_Abandon_TxnCmd(), CF_CFDP_R1_Reset(), CF_CFDP_RecvIdle(), CF_CFDP_S_Reset(), and CF_PurgeTransaction().

Here is the call graph for this function:



```

12.22.3.33 CF_CFDP_SendAck() CFE_Status_t CF_CFDP_SendAck (
    CF_Transaction_t * txn,
    CF_CFDP_AckTxnStatus_t ts,
    CF_CFDP_FileDirective_t dir_code,
    CF_CFDP_ConditionCode_t cc,
    CF_EntityId_t peer_eid,
    CF_TransactionSeq_t tsn )
  
```

Build an ACK PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Note

CF_CFDP_SendAck() takes a CF_TransactionSeq_t instead of getting it from transaction history because of the special case where a FIN-ACK must be sent for an unknown transaction. It's better for long term maintenance to not build an incomplete CF_History_t for it.

Parameters

txn	Pointer to the transaction object
-----	-----------------------------------

Parameters

<i>ts</i>	Transaction ACK status
<i>dir_code</i>	File directive code being ACK'ed
<i>cc</i>	Condition code of transaction
<i>peer_eid</i>	Remote entity ID
<i>tsn</i>	Transaction sequence number

Returns

CFE_Status_t status code

Return values

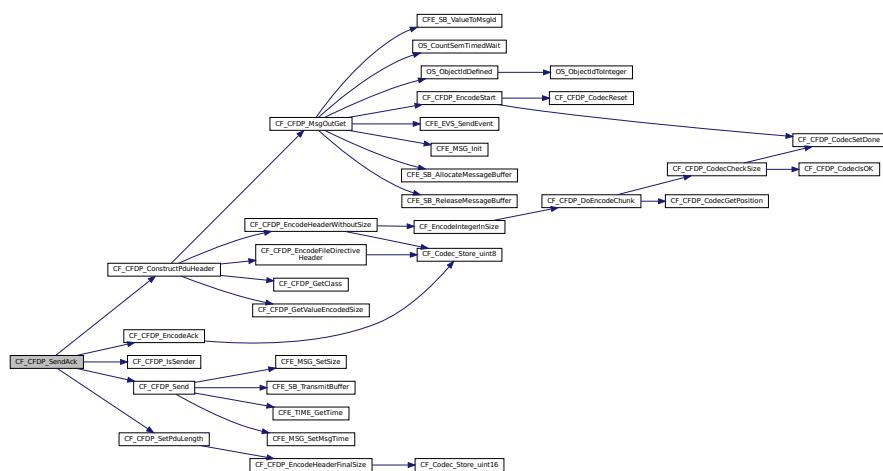
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 447 of file cf_cfdp.c.

References CF_Logical_IntHeader::ack, CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::cc, CF_AppData, CF_ASSERT, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeAck(), CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_EOF, CF_CFDP_FileDirective_FIN, CF_CFDP_IsSender(), CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_Logical_PduBuffer::penc, and CF_Logical_PduAck::txn_status.

Referenced by CF_CFDP_R_Tick(), CF_CFDP_ReceiveMessage(), and CF_CFDP_S_SubstateSendFinAck().

Here is the call graph for this function:



12.22.3.34 CF_CFDP_SendEof() `CFE_Status_t CF_CFDP_SendEof (`
`CF_Transaction_t * txn)`

Build an EOF PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

txn Pointer to the transaction object

Returns

CFE_Status_t status code

Return values

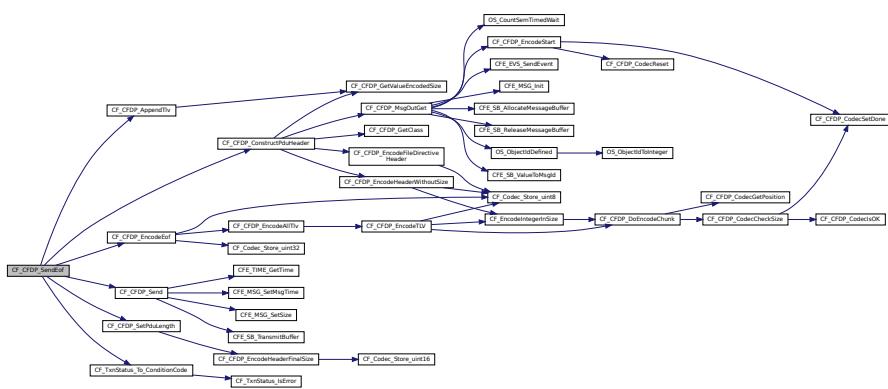
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 408 of file cf_cfdp.c.

References CF_Logical_PduEof::cc, CF_AppData, CF_CFDP_AppendTlv(), CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeEof(), CF_CFDP_FileDirective_EOF, CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_Txn_Status_To_ConditionCode(), CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Logical_PduEof::crc, CF_Transaction::crc, CF_Logical_ExtHeader::eof, CF_Transaction::fsize, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_Crc::result, CF_History::seq_num, CF_Logical_PduEof::size, CF_Logical_PduEof::tlv_list, and CF_History::txn_stat.

Referenced by CF CFDP S SendEof().

Here is the call graph for this function:



12.22.3.35 CF_CFDP_SendEotPkt() void CF_CFDP_SendEotPkt (CF_Transaction_t * txn)

Send an end of transaction packet.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

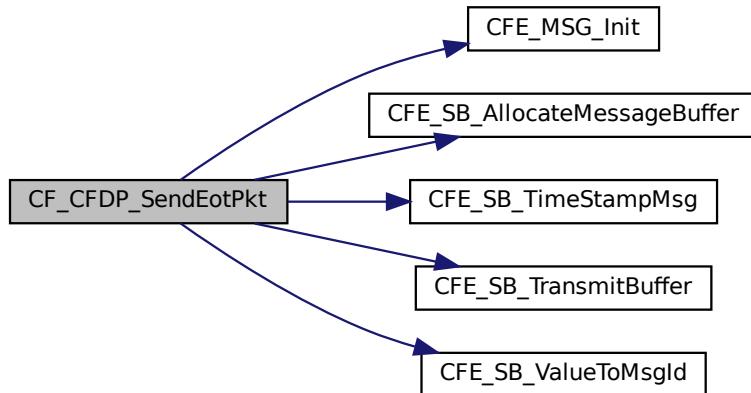
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 1673 of file cf_cfdp.c.

References CF_EOT_TLM_MID, CFE_MSG_Init(), CFE_SB_AllocateMessageBuffer(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitBuffer(), CFE_SB_ValueToMsgId(), CF_Transaction::chan_num, CF_EotPacket_Payload::channel, CF_Transaction::crc, CF_EotPacket_Payload::crc_result, CF_History::dir, CF_EotPacket_Payload::direction, CF_EotPacket_Payload::fnames, CF_History::fnames, CF_EotPacket_Payload::fsize, CF_Transaction::fsize, CF_Transaction::history, CF_EotPacket::Payload, CF_EotPacket_Payload::peer_eid, CF_History::peer_eid, CF_Crc::result, CF_EotPacket_Payload::seq_num, CF_History::seq_num, CF_EotPacket_Payload::src_eid, CF_History::src_eid, CF_EotPacket_Payload::state, CF_Transaction::state, CF_EotPacket::TelemetryHeader, CF_EotPacket_Payload::txn_stat, and CF_History::txn_stat.

Referenced by CF_CFDP_ResetTransaction().

Here is the call graph for this function:



12.22.3.36 CF_CFDP_SendFd() `CF_Status_t CF_CFDP_SendFd (`
 `CF_Transaction_t * txn,`
 `CF_Logical_PduBuffer_t * ph)`

Send a previously-assembled filedata PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

Note

Unlike other "send" routines, the file data PDU must be acquired and filled by the caller prior to invoking this routine. This routine only sends the PDU that was previously allocated and assembled. As such, the typical failure possibilities do not apply to this call.

Returns

CFE_Status_t status code

Return values

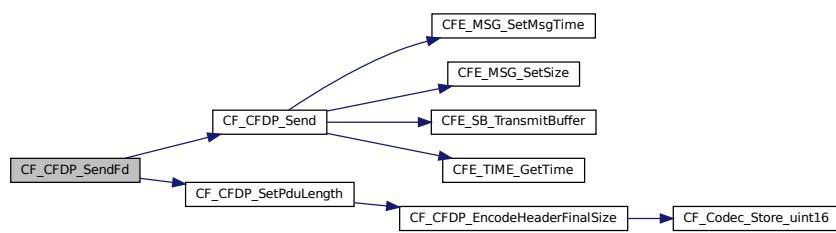
<code>CFE_SUCCESS</code>	on success. (error checks not yet implemented)
--------------------------	--

Definition at line 351 of file cf_cfdp.c.

References CF_CFDP_Send(), CF_CFDP_SetPduLength(), CFE_SUCCESS, and CF_Transaction::chan_num.

Referenced by CF_CFDP_S_SendFileData().

Here is the call graph for this function:

**12.22.3.37 CF_CFDP_SendFin()** `CFE_Status_t CF_CFDP_SendFin(`

```

CF_Transaction_t * txn,
CF_CFDP_FinDeliveryCode_t dc,
CF_CFDP_FinFileStatus_t fs,
CF_CFDP_ConditionCode_t cc )

```

Build a FIN PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
<code>dc</code>	Final delivery status code (complete or incomplete)
<code>fs</code>	Final file status (retained or rejected, etc)
<code>cc</code>	Final CFDP condition code

Returns

`CFE_Status_t` status code

Return values

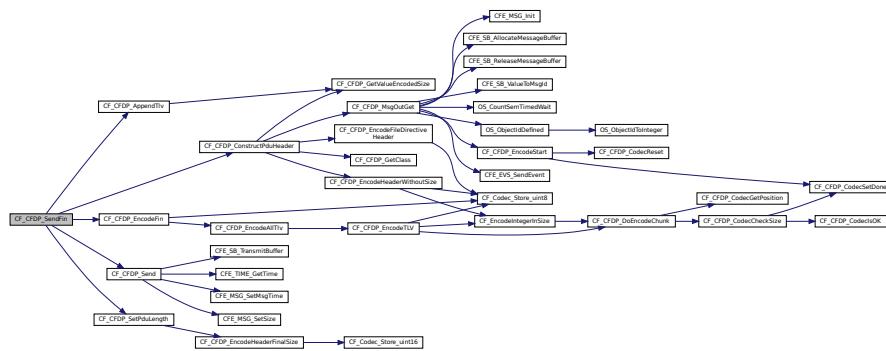
<code>CFE_SUCCESS</code>	on success.
<code>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</code>	if message buffer cannot be obtained.

Definition at line 498 of file cf_cfdp.c.

References `CF_Logical_PduFin::cc`, `CF_AppData`, `CF_CFDP_AppendTlv()`, `CF_CFDP_ConditionCode_NO_ERROR`, `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_EncodeFin()`, `CF_CFDP_FileDirective_FIN`, `CF_CFDP_Send()`, `CF_CFDP_SetPduLength()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_SEND_PDU_NO_BUF_AVAIL_ERROR`, `CFE_SUCCESS`, `CF_Transaction::chan_num`, `CF_AppData_t::config_table`, `CF_Logical_PduFin::delivery_code`, `CF_Logical_PduFin::file_status`, `CF_Logical_ExtHeader::fin`, `CF_Transaction::history`, `CF_Logical_PduBuffer::int_header`, `CF_ConfigTable::local_eid`, `CF_History::peer_eid`, `CF_Logical_PduBuffer::penc`, `CF_History::seq_num`, and `CF_Logical_PduFin::tlv_list`.

Referenced by `CF_CFDP_R2_SubstateSendFin()`.

Here is the call graph for this function:



12.22.3.38 `CF_CFDP_SendMd()` `CFE_Status_t` `CF_CFDP_SendMd(` `CF_Transaction_t * txn)`

Build a metadata PDU for transmit.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Returns

`CFE_Status_t` status code

Return values

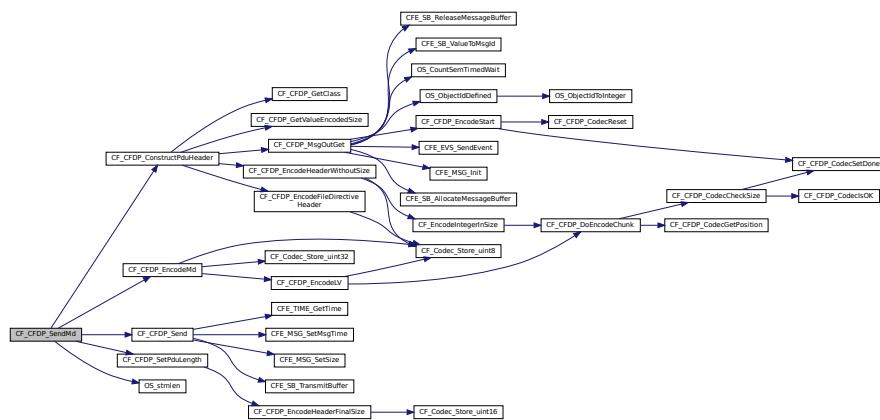
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 308 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_ConstructPduHeader(), CF_CFDP_EncodeMd(), CF_CFDP_File↔Directive_METADATA, CF_CFDP_Send(), CF_CFDP_SetPduLength(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_TxnState_S1, CF_TxnState_S2, CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF↔_Logical_Lv::data_ptr, CF_Logical_PduMd::dest_filename, CF_TxnFilenames::dst_filename, CF_History::fnames, C↔F_Transaction::fsize, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_Logical_Lv::length, CF_Config↔Table::local_eid, CF_Logical_IntHeader::md, OS_strlen(), CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF↔_History::seq_num, CF_Logical_PduMd::size, CF_Logical_PduMd::source_filename, CF_TxnFilenames::src_filename, and CF_Transaction::state.

Referenced by CF_CFDP_S_CheckAndRespondNak(), and CF_CFDP_S_SubstateSendMetadata().

Here is the call graph for this function:



12.22.3.39 CF_CFDP_SendNak() CFE_Status_t CF_CFDP_SendNak (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Send a previously-assembled NAK PDU for transmit.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

Note

Unlike other "send" routines, the NAK PDU must be acquired and filled by the caller prior to invoking this routine. This routine only encodes and sends the previously-assembled PDU buffer. As such, the typical failure possibilities do not apply to this call.

Returns

`CFE_Status_t` status code

Return values

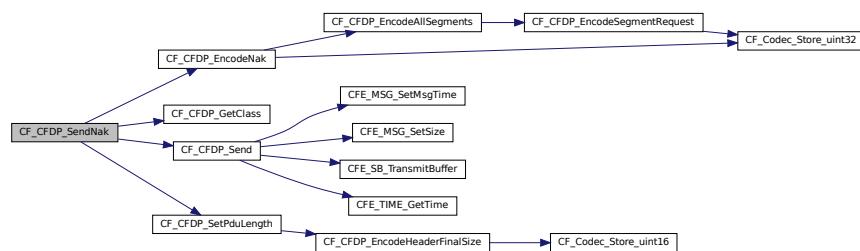
<code>CFE_SUCCESS</code>	on success.
<code>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</code>	if message buffer cannot be obtained.

Definition at line 538 of file cf_cfdp.c.

References `CF_ASSERT`, `CF_CFDP_CLASS_2`, `CF_CFDP_EncodeNak()`, `CF_CFDP_GetClass()`, `CF_CFDP_Send()`, `CF_CFDP_SetPduLength()`, `CF_SEND_PDU_NO_BUF_AVAIL_ERROR`, `CFE_SUCCESS`, `CF_Transaction::chan->num`, `CF_Logical_PduBuffer::int_header`, `CF_Logical_IntHeader::nak`, and `CF_Logical_PduBuffer::penc`.

Referenced by `CF_CFDP_R_SubstateSendNak()`.

Here is the call graph for this function:



12.22.3.40 CF_CFDP_SetTxnStatus() `void CF_CFDP_SetTxnStatus (`
`CF_Transaction_t * txn,`
`CF_TxnStatus_t txn_stat)`

Helper function to store transaction status code only.

This records the status in the history block but does not set FIN flag or take any other protocol/state machine actions.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
<code>txn_stat</code>	Status Code value to set within transaction

Definition at line 1659 of file cf_cfdp.c.

References `CF_TxnStatus_IsError()`, `CF_Transaction::history`, and `CF_History::txn_stat`.

Referenced by CF_CFDP_CancelTransaction(), CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_ProcessFd(), CF_CFDP_R_Tick(), CF_CFDP_RecvFd(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_SubstateSendFileData(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().
Here is the call graph for this function:



12.22.3.41 CF_CFDP_TickTransactions()

```
void CF_CFDP_TickTransactions (
```

```
    CF_Channel_t * chan )
```

Call R and then S tick functions for all active transactions.

Description

Traverses all transactions in the RX and TXW queues, and calls their tick functions. Note that the TXW queue is used twice: once for regular tick processing, and one for NAK response.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

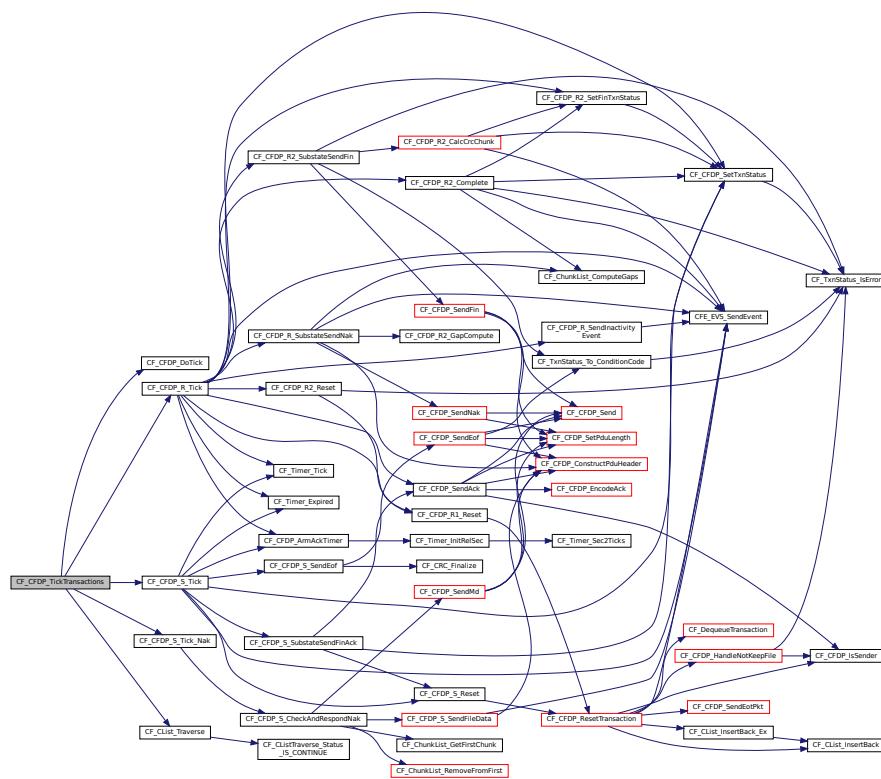
<i>chan</i>	Channel to tick
-------------	-----------------

Definition at line 1153 of file cf_cfdp.c.

References CF_Assert, CF_CFDP_DoTick(), CF_CFDP_R_Tick(), CF_CFDP_S_Tick(), CF_CFDP_S_Tick_Nak(), CF_CList_Traverse(), CF_QueueIdx_RX, CF_QueueIdx_TXW, CF_TickType_NUM_TYPES, CF_TickType_RX, CF_TickType_TXW_NAK, CF_CFDP_Tick_args::cont, CF_CFDP_Tick_args::early_exit, CF_Channel::qs, and CF_Channel::tick_type.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.22.3.42 CF_CFDP_TxFile() CFE_Status_t CF_CFDP_TxFile (

```
const char * src_filename,
const char * dst_filename,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
CF_EntityId_t dest_id )
```

Begin transmit of a file.

Description

This function sets up a transaction for and starts transmit of the given filename.

Assumptions, External Events, and Notes:

src filename must not be NULL. dst filename must not be NULL.

Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename

Parameters

<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
--------------------	--

Returns

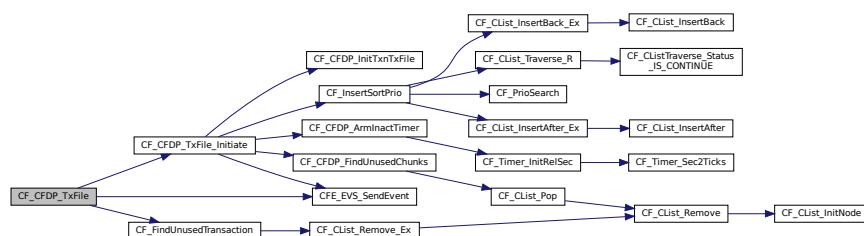
CFE_SUCCESS on success. CF_ERROR on error.

Definition at line 1262 of file cf_cfdp.c.

References CF_AppData, CF_Assert, CF_CFDP_MAX_CMD_TX_ERR_EID, CF_CFDP_TxFile_Initiate(), CF_ER←ROR, CF_FindUnusedTransaction(), CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN, CF_NUM_CHAN←NELS, CF_TxnState_IDLE, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Engine↔::channels, CF_Flags_Tx::cmd_tx, CF_TxnFilenames::dst_filename, CF_AppData_t::engine, CF_Transaction::flags, CF_History::fnames, CF_Transaction::history, CF_Channel::num_cmd_tx, CF_TxnFilenames::src_filename, CF_Transaction::state, and CF_StateFlags::tx.

Referenced by CF_TxFileCmd().

Here is the call graph for this function:



12.23 apps/cf/fsw/src(cf_cfdp_dispatch.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_dispatch.h"
#include <stdio.h>
#include <string.h>
#include "cf_assert.h"
```

Functions

- void `CF_CFDP_R_DispatchRecv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph, const CF_CFDP_R_SubstateDispatchTable_t *dispatch, CF_CFDP_StateRecvFunc_t fd_fn)`
Dispatch function for received PDUs on receive-file transactions.
- void `CF_CFDP_S_DispatchRecv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph, const CF_CFDP_S_SubstateRecvDispatchTable_t *dispatch)`
Dispatch function for received PDUs on send-file transactions.
- void `CF_CFDP_S_DispatchTransmit (CF_Transaction_t *txn, const CF_CFDP_S_SubstateSendDispatchTable_t *dispatch)`
Dispatch function to send/generate PDUs on send-file transactions.
- void `CF_CFDP_TxStateDispatch (CF_Transaction_t *txn, const CF_CFDP_TxnSendDispatchTable_t *dispatch)`
Top-level Dispatch function send a PDU based on current state of a transaction.
- void `CF_CFDP_RxStateDispatch (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph, const CF_CFDP_TxnRecvDispatchTable_t *dispatch)`
Top-level Dispatch function receive a PDU based on current state of a transaction.

12.23.1 Function Documentation

12.23.1.1 CF_CFDP_R_DispatchRecv() void CF_CFDP_R_DispatchRecv (

<code>CF_Transaction_t * txn,</code>	
<code>CF_Logical_PduBuffer_t * ph,</code>	
<code>const CF_CFDP_R_SubstateDispatchTable_t * dispatch,</code>	
<code>CF_CFDP_StateRecvFunc_t fd_fn)</code>	

Dispatch function for received PDUs on receive-file transactions.

Receive file transactions primarily only react/respond to received PDUs

Parameters

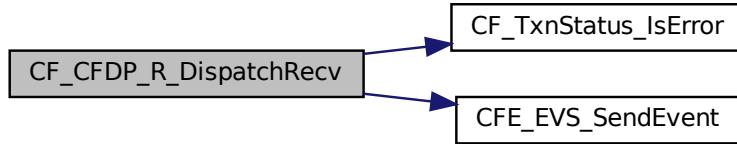
<code>txn</code>	Transaction
<code>ph</code>	PDU Buffer
<code>dispatch</code>	Dispatch table for file directive PDUs
<code>fd_fn</code>	Function to handle file data PDUs

Definition at line 40 of file cf_cfdp_dispatch.c.

References CF_AppData, CF_Assert, CF_CFDP_FileDirective_INVALID_MAX, CF_CFDP_R_DC_INV_ERR_EID, CFE_F_RxSubState_NUM_STATES, CF_TxnState_R2, CF_TxnStatus_IsError(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_HkRecv::dropped, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, CF_Logical_PduBuffer::fdirective, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CFE_F_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_StateData::receive, CF_HkCounters::recv, CF_History::seq_num, CF_HkRecv::spurious, CF_History::src_eid, CF_CFDP_R_SubstateDispatchTable_t::state, CFE_F_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_R1_Recv(), and CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.23.1.2 CF_CFDP_RxStateDispatch() void CF_CFDP_RxStateDispatch (
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph,`
`const CF_CFDP_TxnRecvDispatchTable_t * dispatch)`

Top-level Dispatch function receive a PDU based on current state of a transaction.

Parameters

<i>txn</i>	Transaction
<i>ph</i>	Received PDU Buffer
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 190 of file cf_cfdp_dispatch.c.

References CF_ASSERT, CF_TxnState_INVALID, CF_CFDP_TxnRecvDispatchTable_t::rx, and CF_Transaction::state.
Referenced by CF_CFDP_DispatchRecv().

12.23.1.3 CF_CFDP_S_DispatchRecv() void CF_CFDP_S_DispatchRecv (
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph,`
`const CF_CFDP_S_SubstateRecvDispatchTable_t * dispatch)`

Dispatch function for received PDUs on send-file transactions.

Send file transactions also react/respond to received PDUs. Note that a file data PDU is not expected here.

Parameters

<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs

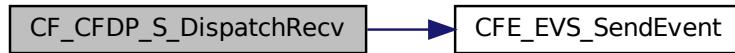
Definition at line 98 of file cf_cfdp_dispatch.c.

References CF_AppData, CF_ASSERT, CF_CFDP_FileDirective_INVALID_MAX, CF_CFDP_S_DC_INV_ERR_EID, CF_CFDP_S_NON_FD_PDU_ERR_EID, CF_TxnState_S2, CF_TxSubState_NUM_STATES, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, CF_Logical_PduBuffer::fdirective, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload,

CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_HkCounters::recv, CF_StateData::send, CF_History::seq_num, CF_HkRecv::spurious, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_TxState_Data::sub_state, and CF_CFDP_S_SubstateRecvDispatchTable_t::substate.

Referenced by CF_CFDP_S1_Recv(), and CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.23.1.4 CF_CFDP_S_DispatchTransmit()

```
void CF_CFDP_S_DispatchTransmit (
    CF_Transaction_t * txn,
    const CF_CFDP_S_SubstateSendDispatchTable_t * dispatch )
```

Dispatch function to send/generate PDUs on send-file transactions.

Send file transactions also generate PDUs each cycle based on the transaction state

This does not have an existing PDU buffer at the time of dispatch, but one may be generated by the invoked function.

Parameters

<i>txn</i>	Transaction
<i>dispatch</i>	State-based dispatch table

Definition at line 155 of file cf_cfdp_dispatch.c.

References CF_StateData::send, CF_Transaction::state_data, CF_TxState_Data::sub_state, and CF_CFDP_S_SubstateSendDispatchTable_t::substate.

Referenced by CF_CFDP_S1_Tx(), and CF_CFDP_S2_Tx().

12.23.1.5 CF_CFDP_TxStateDispatch()

```
void CF_CFDP_TxStateDispatch (
    CF_Transaction_t * txn,
    const CF_CFDP_TxnSendDispatchTable_t * dispatch )
```

Top-level Dispatch function send a PDU based on current state of a transaction.

This does not have an existing PDU buffer at the time of dispatch, but one may be generated by the invoked function.

Parameters

<i>txn</i>	Transaction
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 172 of file cf_cfdp_dispatch.c.

References CF_ASSERT, CF_TxnState_INVALID, CF_Transaction::state, and CF_CFDP_TxnSendDispatchTable_t::tx. Referenced by CF_CFDP_DispatchTx().

12.24 apps/cf/fsw/src(cf_cfdp_dispatch.h File Reference

```
#include "cf_cfdp_types.h"
```

Data Structures

- struct [CF_CFDP_TxnSendDispatchTable_t](#)
A table of transmit handler functions based on transaction state.
- struct [CF_CFDP_TxnRecvDispatchTable_t](#)
A table of receive handler functions based on transaction state.
- struct [CF_CFDP_FileDirectiveDispatchTable_t](#)
A table of receive handler functions based on file directive code.
- struct [CF_CFDP_R_SubstateDispatchTable_t](#)
A dispatch table for receive file transactions, receive side.
- struct [CF_CFDP_S_SubstateRecvDispatchTable_t](#)
A dispatch table for send file transactions, receive side.
- struct [CF_CFDP_S_SubstateSendDispatchTable_t](#)
A dispatch table for send file transactions, transmit side.

Typedefs

- typedef void(* [CF_CFDP_StateSendFunc_t](#)) ([CF_Transaction_t](#) *txn)
A function for dispatching actions to a handler, without existing PDU data.
- typedef void(* [CF_CFDP_StateRecvFunc_t](#)) ([CF_Transaction_t](#) *txn, [CF_Logical_PduBuffer_t](#) *ph)
A function for dispatching actions to a handler, with existing PDU data.

Functions

- void [CF_CFDP_R_DispatchRecv](#) ([CF_Transaction_t](#) *txn, [CF_Logical_PduBuffer_t](#) *ph, const [CF_CFDP_R_SubstateDispatchTable_t](#) *dispatch, [CF_CFDP_StateRecvFunc_t](#) fd_fn)
Dispatch function for received PDUs on receive-file transactions.
- void [CF_CFDP_S_DispatchRecv](#) ([CF_Transaction_t](#) *txn, [CF_Logical_PduBuffer_t](#) *ph, const [CF_CFDP_S_SubstateRecvDispatchTable_t](#) *dispatch)
Dispatch function for received PDUs on send-file transactions.
- void [CF_CFDP_S_DispatchTransmit](#) ([CF_Transaction_t](#) *txn, const [CF_CFDP_S_SubstateSendDispatchTable_t](#) *dispatch)
Dispatch function to send/generate PDUs on send-file transactions.
- void [CF_CFDP_TxStateDispatch](#) ([CF_Transaction_t](#) *txn, const [CF_CFDP_TxnSendDispatchTable_t](#) *dispatch)
Top-level Dispatch function send a PDU based on current state of a transaction.
- void [CF_CFDP_RxStateDispatch](#) ([CF_Transaction_t](#) *txn, [CF_Logical_PduBuffer_t](#) *ph, const [CF_CFDP_TxnRecvDispatchTable_t](#) *dispatch)
Top-level Dispatch function receive a PDU based on current state of a transaction.

12.24.1 Detailed Description

Common routines to dispatch operations based on a transaction state and/or received PDU type.

12.24.2 Typedef Documentation

12.24.2.1 CF_CFDP_StateRecvFunc_t `typedef void(* CF_CFDP_StateRecvFunc_t) (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`

A function for dispatching actions to a handler, with existing PDU data.

This allows quick delegation of PDUs to handler functions using dispatch tables. This version is used on the receive side where a PDU buffer is associated with the activity, which is then interpreted by the handler being invoked.

Parameters

in, out	<i>txn</i>	The transaction object
in, out	<i>ph</i>	The PDU buffer currently being received/processed

Definition at line 53 of file cf_cfdp_dispatch.h.

12.24.2.2 CF_CFDP_StateSendFunc_t `typedef void(* CF_CFDP_StateSendFunc_t) (CF_Transaction_t *txn)`

A function for dispatching actions to a handler, without existing PDU data.

This allows quick delegation to handler functions using dispatch tables. This version is used on the transmit side, where a PDU will likely be generated/sent by the handler being invoked.

Parameters

in, out	<i>txn</i>	The transaction object
---------	------------	------------------------

Definition at line 41 of file cf_cfdp_dispatch.h.

12.24.3 Function Documentation

12.24.3.1 CF_CFDP_R_DispatchRecv() `void CF_CFDP_R_DispatchRecv (`

`CF_Transaction_t * txn,`

`CF_Logical_PduBuffer_t * ph,`

`const CF_CFDP_R_SubstateDispatchTable_t * dispatch,`

`CF_CFDP_StateRecvFunc_t fd_fn)`

Dispatch function for received PDUs on receive-file transactions.

Receive file transactions primarily only react/respond to received PDUs

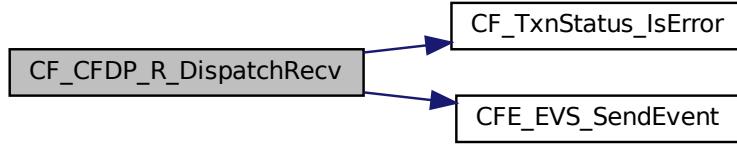
Parameters

<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs
<i>fd_fn</i>	Function to handle file data PDUs

Definition at line 40 of file cf_cfdp_dispatch.c.

References CF_AppData, CF_ASSERT, CF_CFDP_FileDirective_INVALID_MAX, CF_CFDP_R_DC_INV_ERR_EID, CFE_RxSubState_NUM_STATES, CF_TxnState_R2, CF_TxnStatus_IsError(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_HkRecv::dropped, CF_CFDP_FileDirectiveDispatchTable::t::fdirective, CF_Logical_PduBuffer::fdirective, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_StateData::receive, CF_HkCounters::recv,

CF_History::seq_num, CF_HkRecv::spurious, CF_History::src_eid, CF_CFDP_R_SubstateDispatchTable_t::state, C← F_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.
Referenced by CF_CFDP_R1_Recv(), and CF_CFDP_R2_Recv().
Here is the call graph for this function:



12.24.3.2 CF_CFDP_RxStateDispatch() void CF_CFDP_RxStateDispatch (
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph,`
`const CF_CFDP_TxnRecvDispatchTable_t * dispatch)`

Top-level Dispatch function receive a PDU based on current state of a transaction.

Parameters

<i>txn</i>	Transaction
<i>ph</i>	Received PDU Buffer
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 190 of file cf_cfdp_dispatch.c.

References CF_ASSERT, CF_TxnState_INVALID, CF_CFDP_TxnRecvDispatchTable_t::rx, and CF_Transaction::state.
Referenced by CF_CFDP_DispatchRecv().

12.24.3.3 CF_CFDP_S_DispatchRecv() void CF_CFDP_S_DispatchRecv (
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph,`
`const CF_CFDP_S_SubstateRecvDispatchTable_t * dispatch)`

Dispatch function for received PDUs on send-file transactions.

Send file transactions also react/respond to received PDUs. Note that a file data PDU is not expected here.

Parameters

<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs

Definition at line 98 of file cf_cfdp_dispatch.c.

References CF_AppData, CF_ASSERT, CF_CFDP_FileDirective_INVALID_MAX, CF_CFDP_S_DC_INV_ERR_EID, C← F_CFDP_S_NON_FD_PDU_ERR_EID, CF_TxnState_S2, CF_TxSubState_NUM_STATES, CFE_EVS_EventType←

ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel<→_Data::counters, CF_Logical_PduFileDirectiveHeader::directive_code, CF_CFDP_FileDirectiveDispatchTable_t<→::fdirective, CF_Logical_PduBuffer::fdirective, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_Logical_PduHeader::pdu_type, CF_HkCounters::recv, CF_StateData::send, CF_History::seq_num, CF_HkRecv::spurious, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_TxState_Data::sub_state, and CF_CFDP_S_SubstateRecvDispatchTable_t::substate.

Referenced by CF_CFDP_S1_Recv(), and CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.24.3.4 CF_CFDP_S_DispatchTransmit() void CF_CFDP_S_DispatchTransmit (
 CF_Transaction_t * *txn*,
 const CF_CFDP_S_SubstateSendDispatchTable_t * *dispatch*)

Dispatch function to send/generate PDUs on send-file transactions.

Send file transactions also generate PDUs each cycle based on the transaction state

This does not have an existing PDU buffer at the time of dispatch, but one may be generated by the invoked function.

Parameters

<i>txn</i>	Transaction
<i>dispatch</i>	State-based dispatch table

Definition at line 155 of file cf_cfdp_dispatch.c.

References CF_StateData::send, CF_Transaction::state_data, CF_TxState_Data::sub_state, and CF_CFDP_S<→_SubstateSendDispatchTable_t::substate.

Referenced by CF_CFDP_S1_Tx(), and CF_CFDP_S2_Tx().

12.24.3.5 CF_CFDP_TxStateDispatch() void CF_CFDP_TxStateDispatch (
 CF_Transaction_t * *txn*,
 const CF_CFDP_TxnSendDispatchTable_t * *dispatch*)

Top-level Dispatch function send a PDU based on current state of a transaction.

This does not have an existing PDU buffer at the time of dispatch, but one may be generated by the invoked function.

Parameters

<i>txn</i>	Transaction
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 172 of file cf_cfdp_dispatch.c.

References CF_ASSERT, CF_TxnState_INVALID, CF_Transaction::state, and CF_CFDP_TxnSendDispatchTable_t::tx. Referenced by CF_CFDP_DispatchTx().

12.25 apps/cf/fsw/src(cf_cfdp_pdu.h File Reference

```
#include "common_types.h"
#include "cf_platform_cfg.h"
#include <stddef.h>
```

Data Structures

- struct [CF_CFDP_uint8_t](#)
Encoded 8-bit value in the CFDP PDU.
- struct [CF_CFDP_uint16_t](#)
Encoded 16-bit value in the CFDP PDU.
- struct [CF_CFDP_uint32_t](#)
Encoded 32-bit value in the CFDP PDU.
- struct [CF_CFDP_uint64_t](#)
Encoded 64-bit value in the CFDP PDU.
- struct [CF_CFDP_PduHeader](#)
Structure representing base CFDP PDU header.
- struct [CF_CFDP_PduFileDirectiveHeader](#)
Structure representing CFDP File Directive Header.
- struct [CF_CFDP_lv](#)
Structure representing CFDP LV Object format.
- struct [CF_CFDP_tlv](#)
Structure representing CFDP TLV Object format.
- struct [CF_CFDP_PduEof](#)
Structure representing CFDP End of file PDU.
- struct [CF_CFDP_PduFin](#)
Structure representing CFDP Finished PDU.
- struct [CF_CFDP_PduAck](#)
Structure representing CFDP Acknowledge PDU.
- struct [CF_CFDP_SegmentRequest](#)
Structure representing CFDP Segment Request.
- struct [CF_CFDP_PduNak](#)
Structure representing CFDP Non-Acknowledge PDU.
- struct [CF_CFDP_PduMd](#)
Structure representing CFDP Metadata PDU.
- struct [CF_CFDP_PduFileDataHeader](#)
PDU file data header.
- struct [CF_CFDP_PduFileDataContent](#)
PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command.

Macros

- #define [CF_CFDP_MAX_HEADER_SIZE](#) (sizeof([CF_CFDP_PduHeader_t](#)) + (3 * sizeof([CF_CFDP_uint64_t](#))))
/ 8 bytes for each variable item */*
Maximum encoded size of a CFDP PDU header.
- #define [CF_CFDP_MIN_HEADER_SIZE](#) (sizeof([CF_CFDP_PduHeader_t](#)) + (3 * sizeof([CF_CFDP_uint8_t](#)))) /*
*1 byte for each variable item */*

Minimum encoded size of a CFDP PDU header.

- #define **CF_APP_MAX_HEADER_SIZE** (sizeof(**CF_CFDP_PduHeader_t**) + sizeof(**CF_TransactionSeq_t**) + (3 * sizeof(**CF_EntityId_t**)))

Maximum encoded size of a CFDP PDU that this implementation can accept.

Typedefs

- typedef struct **CF_CFDP_PduHeader** **CF_CFDP_PduHeader_t**
Structure representing base CFDP PDU header.
- typedef struct **CF_CFDP_PduFileDirectiveHeader** **CF_CFDP_PduFileDirectiveHeader_t**
Structure representing CFDP File Directive Header.
- typedef struct **CF_CFDP_lv** **CF_CFDP_lv_t**
Structure representing CFDP LV Object format.
- typedef struct **CF_CFDP_tlv** **CF_CFDP_tlv_t**
Structure representing CFDP TLV Object format.
- typedef struct **CF_CFDP_PduEof** **CF_CFDP_PduEof_t**
Structure representing CFDP End of file PDU.
- typedef struct **CF_CFDP_PduFin** **CF_CFDP_PduFin_t**
Structure representing CFDP Finished PDU.
- typedef struct **CF_CFDP_PduAck** **CF_CFDP_PduAck_t**
Structure representing CFDP Acknowledge PDU.
- typedef struct **CF_CFDP_SegmentRequest** **CF_CFDP_SegmentRequest_t**
Structure representing CFDP Segment Request.
- typedef struct **CF_CFDP_PduNak** **CF_CFDP_PduNak_t**
Structure representing CFDP Non-Acknowledge PDU.
- typedef struct **CF_CFDP_PduMd** **CF_CFDP_PduMd_t**
Structure representing CFDP Metadata PDU.
- typedef struct **CF_CFDP_PduFileDataHeader** **CF_CFDP_PduFileDataHeader_t**
PDU file data header.
- typedef struct **CF_CFDP_PduFileDataContent** **CF_CFDP_PduFileDataContent_t**
PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command.

Enumerations

- enum **CF_CFDP_TlvType_t**{
 CF_CFDP_TLV_TYPE_FILESTORE_REQUEST = 0, **CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE** = 1,
 CF_CFDP_TLV_TYPE_MESSAGE_TO_USER = 2, **CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE** =
 4,
 CF_CFDP_TLV_TYPE_FLOW_LABEL = 5, **CF_CFDP_TLV_TYPE_ENTITY_ID** = 6, **CF_CFDP_TLV_TYPE_INVALID_MAX**
 = 7 }
Values for "type" field of TLV structure.
- enum **CF_CFDP_FileDirective_t**{
 CF_CFDP_FileDirective_INVALID_MIN = 0, **CF_CFDP_FileDirective_EOF** = 4, **CF_CFDP_FileDirective_FIN** =
 5, **CF_CFDP_FileDirective_ACK** = 6,
 CF_CFDP_FileDirective_METADATA = 7, **CF_CFDP_FileDirective_NAK** = 8, **CF_CFDP_FileDirective_PROMPT**
 = 9, **CF_CFDP_FileDirective_KEEP_ALIVE** = 12,
 CF_CFDP_FileDirective_INVALID_MAX = 13 }
Values for "directive_code" within CF_CFDP_PduFileDirectiveHeader_t.

- enum `CF_CFDP_AckTxnStatus_t` {
 `CF_CFDP_AckTxnStatus_UNDEFINED` = 0, `CF_CFDP_AckTxnStatus_ACTIVE` = 1, `CF_CFDP_AckTxnStatus_TERMINATED` = 2, `CF_CFDP_AckTxnStatus_UNRECOGNIZED` = 3,
`CF_CFDP_AckTxnStatus_INVALID` = 4 }

Values for "acknowledgment transfer status".
- enum `CF_CFDP_FinDeliveryCode_t` { `CF_CFDP_FinDeliveryCode_COMPLETE` = 0, `CF_CFDP_FinDeliveryCode_INCOMPLETE` = 1, `CF_CFDP_FinDeliveryCode_INVALID` = 2 }

Values for "finished delivery code".
- enum `CF_CFDP_FinFileStatus_t` {
 `CF_CFDP_FinFileStatus_DISCARDED` = 0, `CF_CFDP_FinFileStatus_DISCARDED_FILESTORE` = 1,
`CF_CFDP_FinFileStatus_RETAINED` = 2, `CF_CFDP_FinFileStatus_UNREPORTED` = 3,
`CF_CFDP_FinFileStatus_INVALID` = 4 }

Values for "finished file status".
- enum `CF_CFDP_ConditionCode_t` {
 `CF_CFDP_ConditionCode_NO_ERROR` = 0, `CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED` = 1,
`CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED` = 2, `CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE` = 3,
`CF_CFDP_ConditionCode_FILESTORE_REJECTION` = 4, `CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE` = 5, `CF_CFDP_ConditionCode_FILE_SIZE_ERROR` = 6, `CF_CFDP_ConditionCode_NAK_LIMIT_REACHED` = 7,
`CF_CFDP_ConditionCode_INACTIVITY_DETECTED` = 8, `CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE` = 9, `CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED` = 10, `CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE` = 11,
`CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED` = 14, `CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED` = 15 }

Values for "condition code".

12.25.1 Detailed Description

Structures defining to CFDP PDUs

Note that structures and enumerations defined in this file with a CF_CFDP prefix are defined according to the CCSDS CFDP specification (727.0-B-5). These values must match the specification for that structure/field, they are not locally changeable.

Note

Many of the structures defined in this file are variably-sized when encoded for network transmission. As a result, C structures used to map to these structures are of limited usefulness, generally only capable of describing the first element(s) where offsets are fixed. A marker member is utilized to indicate where the fixed data ends and variable length data begins. At some point, the structures in this file should change to encode/decode functions.

12.25.2 Macro Definition Documentation

12.25.2.1 CF_APP_MAX_HEADER_SIZE #define CF_APP_MAX_HEADER_SIZE (sizeof(`CF_CFDP_PduHeader_t`) + sizeof(`CF_TransactionSeq_t`) + (3 * sizeof(`CF_EntityId_t`)))

Maximum encoded size of a CFDP PDU that this implementation can accept.

This definition reflects the current configuration of the CF application. Note that this is based on the size of the native representation of Entity ID and sequence number. Although the bitwise representations of these items are different in the encoded packets vs. the native representation, the basic size still correlates (e.g. if it takes 4 bytes natively, it will be encoded into 4 bytes).

Definition at line 75 of file cf_cfdp_pdu.h.

12.25.2.2 CF_CFDP_MAX_HEADER_SIZE #define CF_CFDP_MAX_HEADER_SIZE (sizeof(CF_CFDP_PduHeader_t) + (3 * sizeof(CF_CFDP_uint64_t))) /* 8 bytes for each variable item */

Maximum encoded size of a CFDP PDU header.

Per the blue book, the size of the Entity ID and Sequence Number may be up to 8 bytes. CF is configurable in what it can accept and transmit, which may be smaller than what the blue book permits.

Definition at line 54 of file cf_cfdp_pdu.h.

12.25.2.3 CF_CFDP_MIN_HEADER_SIZE #define CF_CFDP_MIN_HEADER_SIZE (sizeof(CF_CFDP_PduHeader_t) + (3 * sizeof(CF_CFDP_uint8_t))) /* 1 byte for each variable item */

Minimum encoded size of a CFDP PDU header.

Per the blue book, the size of the Entity ID and Sequence Number must be at least 1 byte.

Definition at line 62 of file cf_cfdp_pdu.h.

12.25.3 Typedef Documentation

12.25.3.1 CF_CFDP_lv_t typedef struct CF_CFDP_lv CF_CFDP_lv_t

Structure representing CFDP LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

Defined per table 5-2 of CCSDS 727.0-B-5

12.25.3.2 CF_CFDP_PduAck_t typedef struct CF_CFDP_PduAck CF_CFDP_PduAck_t

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

12.25.3.3 CF_CFDP_PduEof_t typedef struct CF_CFDP_PduEof CF_CFDP_PduEof_t

Structure representing CFDP End of file PDU.

Defined per section 5.2.2 / table 5-6 of CCSDS 727.0-B-5

12.25.3.4 CF_CFDP_PduFileDataContent_t typedef struct CF_CFDP_PduFileDataContent CF_CFDP_PduFileDataContent_t

PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command.

This definition allows for the largest data block possible, as CF_MAX_PDU_SIZE - the minimum possible header size.

In practice the outgoing file chunk size is limited by whichever is smaller; the remaining data, remaining space in the packet, and outgoing_file_chunk_size.

12.25.3.5 CF_CFDP_PduFileDataHeader_t typedef struct CF_CFDP_PduFileDataHeader CF_CFDP_PduFileDataHeader_t

PDU file data header.

12.25.3.6 CF_CFDP_PduFileDirectiveHeader_t typedef struct CF_CFDP_PduFileDirectiveHeader CF_CFDP_PduFileDirectiveHeader_t

Structure representing CFDP File Directive Header.

Defined per section 5.2 of CCSDS 727.0-B-5

12.25.3.7 CF_CFDP_PduFin_t typedef struct CF_CFDP_PduFin CF_CFDP_PduFin_t

Structure representing CFDP Finished PDU.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

12.25.3.8 CF_CFDP_PduHeader_t `typedef struct CF_CFDP_PduHeader CF_CFDP_PduHeader_t`

Structure representing base CFDP PDU header.

This header appears at the beginning of all CFDP PDUs, of all types. Note that the header is variable length, it also contains source and destination entity IDs, and the transaction sequence number.

Defined per section 5.1 of CCSDS 727.0-B-5

Note

this contains variable length data for the EID+TSN, which is *not* included in this definition. As a result, the sizeof(`CF_CFDP_PduHeader_t`) reflects only the size of the fixed fields. Use `CF_HeaderSize()` to get the actual size of this structure.

12.25.3.9 CF_CFDP_PduMd_t `typedef struct CF_CFDP_PduMd CF_CFDP_PduMd_t`

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

12.25.3.10 CF_CFDP_PduNak_t `typedef struct CF_CFDP_PduNak CF_CFDP_PduNak_t`

Structure representing CFDP Non-Acknowledge PDU.

Defined per section 5.2.6 / table 5-10 of CCSDS 727.0-B-5

12.25.3.11 CF_CFDP_SegmentRequest_t `typedef struct CF_CFDP_SegmentRequest CF_CFDP_SegmentRequest_t`

Structure representing CFDP Segment Request.

Defined per section 5.2.6 / table 5-11 of CCSDS 727.0-B-5

12.25.3.12 CF_CFDP_tlv_t `typedef struct CF_CFDP_tlv CF_CFDP_tlv_t`

Structure representing CFDP TLV Object format.

These Type + Length + Value pairs used in several CFDP PDU types, typically for file storage requests (section 5.4).

Defined per table 5-3 of CCSDS 727.0-B-5

12.25.4 Enumeration Type Documentation

12.25.4.1 CF_CFDP_AckTxnStatus_t `enum CF_CFDP_AckTxnStatus_t`

Values for "acknowledgment transfer status".

This enum is pertinent to the ACK PDU type, defines the values for the directive field.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Enumerator

<code>CF_CFDP_AckTxnStatus_UNDEFINED</code>	
<code>CF_CFDP_AckTxnStatus_ACTIVE</code>	
<code>CF_CFDP_AckTxnStatus_TERMINATED</code>	
<code>CF_CFDP_AckTxnStatus_UNRECOGNIZED</code>	
<code>CF_CFDP_AckTxnStatus_INVALID</code>	

Definition at line 225 of file cf_cfdp_pdu.h.

12.25.4.2 CF_CFDP_ConditionCode_t `enum CF_CFDP_ConditionCode_t`

Values for "condition code".

This enum defines the values for the condition code field for the PDU types which have this field (EOF, FIN, ACK)
Defined per table 5-5 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_ConditionCode_NO_ERROR
CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED
CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED
CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE
CF_CFDP_ConditionCode_FILESTORE_REJECTION
CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE
CF_CFDP_ConditionCode_FILE_SIZE_ERROR
CF_CFDP_ConditionCode_NAK_LIMIT_REACHED
CF_CFDP_ConditionCode_INACTIVITY_DETECTED
CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE
CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED
CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE
CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED
CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED

Definition at line 274 of file cf_cfdp_pdu.h.

12.25.4.3 CF_CFDP_FileDirective_t enum [CF_CFDP_FileDirective_t](#)

Values for "directive_code" within CF_CFDP_PduFileDirectiveHeader_t.

Defined per table 5-4 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_FileDirective_INVALID_MIN	Minimum used to limit range.
CF_CFDP_FileDirective_EOF	
CF_CFDP_FileDirective_FIN	
CF_CFDP_FileDirective_ACK	
CF_CFDP_FileDirective_METADATA	
CF_CFDP_FileDirective_NAK	
CF_CFDP_FileDirective_PROMPT	
CF_CFDP_FileDirective_KEEP_ALIVE	
CF_CFDP_FileDirective_INVALID_MAX	Maximum used to limit range.

Definition at line 204 of file cf_cfdp_pdu.h.

12.25.4.4 CF_CFDP_FinDeliveryCode_t enum [CF_CFDP_FinDeliveryCode_t](#)

Values for "finished delivery code".

This enum is pertinent to the FIN PDU type, defines the values for the delivery code field.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_FinDeliveryCode_COMPLETE

Enumerator

CF_CFDP_FinDeliveryCode_INCOMPLETE	
CF_CFDP_FinDeliveryCode_INVALID	

Definition at line 242 of file cf_cfdp_pdu.h.

12.25.4.5 CF_CFDP_FinFileStatus_t enum CF_CFDP_FinFileStatus_t

Values for "finished file status".

This enum is pertinent to the FIN PDU type, defines the values for the file status field.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_FinFileStatus_DISCARDED	
CF_CFDP_FinFileStatus_DISCARDED_FILESTORE	
CF_CFDP_FinFileStatus_RETAINED	
CF_CFDP_FinFileStatus_UNREPORTED	
CF_CFDP_FinFileStatus_INVALID	

Definition at line 257 of file cf_cfdp_pdu.h.

12.25.4.6 CF_CFDP_TlvType_t enum CF_CFDP_TlvType_t

Values for "type" field of TLV structure.

Defined per section 5.4 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_TLV_TYPE_FILESTORE_REQUEST	
CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE	
CF_CFDP_TLV_TYPE_MESSAGE_TO_USER	
CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE	
CF_CFDP_TLV_TYPE_FLOW_LABEL	
CF_CFDP_TLV_TYPE_ENTITY_ID	
CF_CFDP_TLV_TYPE_INVALID_MAX	

Definition at line 188 of file cf_cfdp_pdu.h.

12.26 apps/cf/fsw/src(cf_cfdp_r.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_dispatch.h"
```

```
#include <stdio.h>
#include <string.h>
#include "cf_assert.h"
```

Functions

- void **CF_CFDP_R2_SetFinTxnStatus** (CF_Transaction_t *txn, CF_TxnStatus_t txn_stat)
Helper function to store transaction status code and set send_fin flag.
- void **CF_CFDP_R1_Reset** (CF_Transaction_t *txn)
CFDP R1 transaction reset function.
- void **CF_CFDP_R2_Reset** (CF_Transaction_t *txn)
CFDP R2 transaction reset function.
- CFE_Status_t **CF_CFDP_R_CheckCrc** (CF_Transaction_t *txn, uint32 expected_crc)
Checks that the transaction file's CRC matches expected.
- void **CF_CFDP_R2_Complete** (CF_Transaction_t *txn, int ok_to_send_nak)
Checks R2 transaction state for transaction completion status.
- CFE_Status_t **CF_CFDP_R_ProcessFd** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process a filedata PDU on a transaction.
- CFE_Status_t **CF_CFDP_R_SubstateRecvEof** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Processing receive EOF common functionality for R1/R2.
- void **CF_CFDP_R1_SubstateRecvEof** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process receive EOF for R1.
- void **CF_CFDP_R2_SubstateRecvEof** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process receive EOF for R2.
- void **CF_CFDP_R1_SubstateRecvFileData** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process received file data for R1.
- void **CF_CFDP_R2_SubstateRecvFileData** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process received file data for R2.
- void **CF_CFDP_R2_GapCompute** (const CF_ChunkList_t *chunks, const CF_Chunk_t *chunk, void *opaque)
Loads a single NAK segment request.
- CFE_Status_t **CF_CFDP_R_SubstateSendNak** (CF_Transaction_t *txn)
Send a NAK PDU for R2.
- void **CF_CFDP_R_Init** (CF_Transaction_t *txn)
Initialize a transaction structure for R.
- CFE_Status_t **CF_CFDP_R2_CalcCrcChunk** (CF_Transaction_t *txn)
Calculate up to the configured amount of bytes of CRC.
- CFE_Status_t **CF_CFDP_R2_SubstateSendFin** (CF_Transaction_t *txn)
Send a FIN PDU.
- void **CF_CFDP_R2_Recv_fin_ack** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process receive FIN-ACK PDU.
- void **CF_CFDP_R2_RecvMd** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
Process receive metadata PDU for R2.
- void **CF_CFDP_R1_Recv** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
R1 receive PDU processing.
- void **CF_CFDP_R2_Recv** (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)
R2 receive PDU processing.
- void **CF_CFDP_R_Cancel** (CF_Transaction_t *txn)

Cancel an R transaction.

- void [CF_CFDP_R_SendInactivityEvent](#) ([CF_Transaction_t](#) **txn*)
Sends an inactivity timer expired event to EVS.
- void [CF_CFDP_R_Tick](#) ([CF_Transaction_t](#) **txn*, int **cont*)
Perform tick (time-based) processing for R transactions.

12.26.1 Detailed Description

The CF Application CFDP receive logic source file
Handles all CFDP engine functionality specific to RX transactions.

12.26.2 Function Documentation

12.26.2.1 CF_CFDP_R1_Recv() void CF_CFDP_R1_Recv (

CF_Transaction_t *	<i>txn</i> ,
CF_Logical_PduBuffer_t *	<i>ph</i>)

R1 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

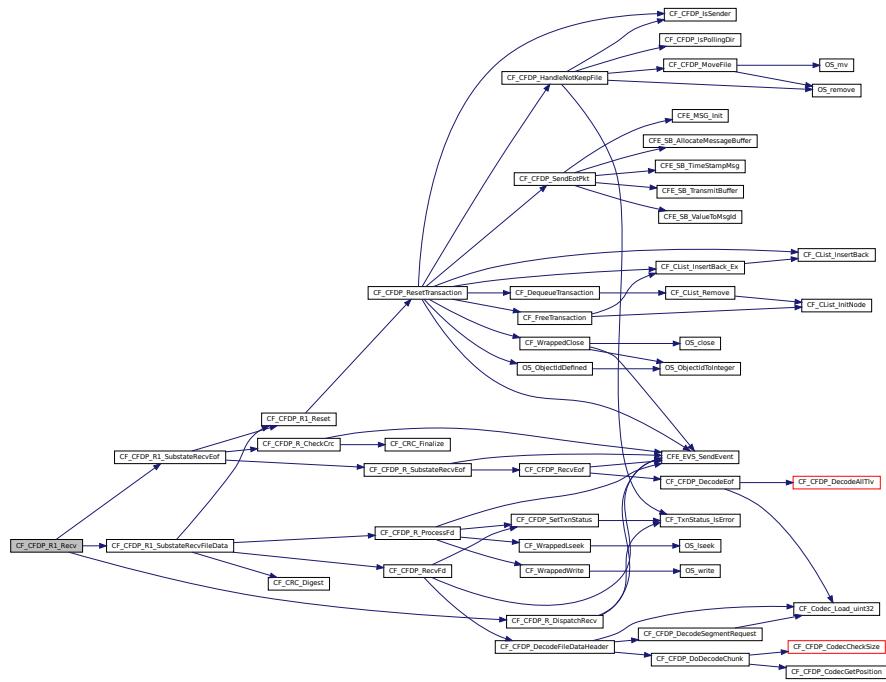
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 882 of file cf_cfdp_r.c.

References [CF_CFDP_FileDirective_EOF](#), [CF_CFDP_R1_SubstateRecvEof\(\)](#), [CF_CFDP_R1_SubstateRecvFileData\(\)](#), [CF_CFDP_R_DispatchRecv\(\)](#), [CF_RxSubState_EOF](#), [CF_RxSubState_FILENODATA](#), [CF_RxSubState_WAIT_FOR_FIN_ACK](#), [CF_CFDP_FileDirectiveDispatchTable_t::fdirective](#), and [CF_CFDP_R_SubstateDispatchTable_t::state](#).

Referenced by [CF_CFDP_DispatchRecv\(\)](#).

Here is the call graph for this function:



12.26.2.2 CF_CFDP_R1_Reset() void CF_CFDP_R1_Reset (

CFDP R1 transaction reset function.

Description

All R transactions use this call to indicate the transaction state can be returned to the system. While this function currently only calls [CF_CFDP_ResetTransaction\(\)](#), it is here as a placeholder.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

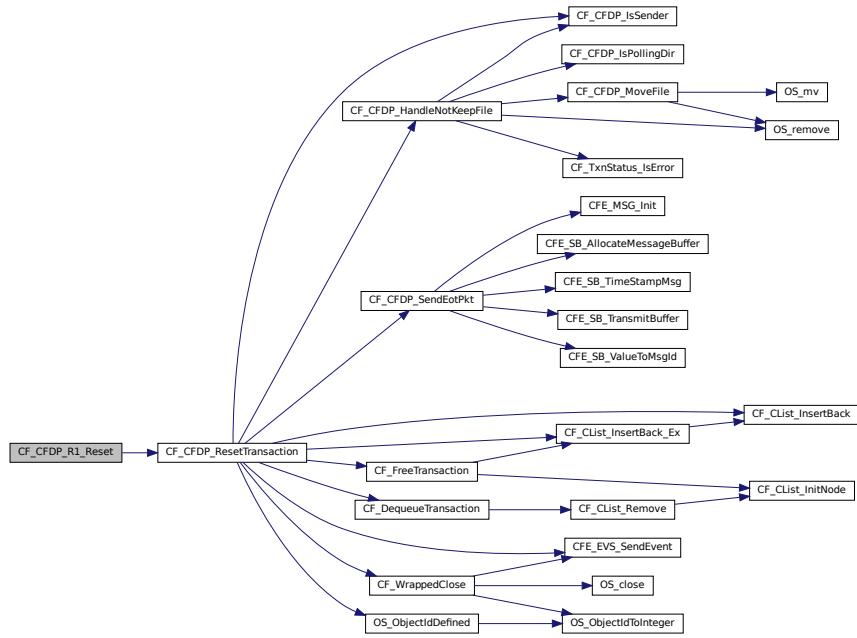
txn Pointer to the transaction object

Definition at line 59 of file cf_cfdp_r.c.

References CF_CFDP_ResetTransaction().

Referenced by CF_CFDP_R1_SubstateRecvEof(), CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R2_Reset(), CE_CEDP_B_Cancel(), CE_CEDP_B_Init(), and CE_CEDP_B_Tick().

Here is the call graph for this function:



12.26.2.3 CF_CFDP_R1_SubstateRecvEof() void CF_CFDP_R1_SubstateRecvEof (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

Process receive EOF for R1.

Description

Only need to confirm CRC for R1.

Assumptions, External Events, and Notes:

tx must not be NULL. ph must not be NULL.

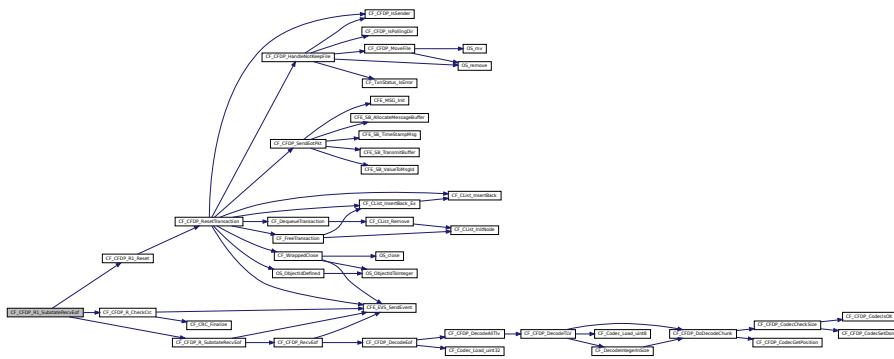
Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 290 of file cf_cfdp_r.c.

References `CF_CFDP_R1_Reset()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_SubstateRecvEof()`, `CFE_SUCCESS`, `CF_Logical_PduEof::crc`, `CF_Logical_ImgHeader::eof`, `CF_Logical_PduBuffer::int_header`, and `CF_Transaction::keep`. Referenced by `CF_CFDP_R1_Recv()`.

Here is the call graph for this function:



12.26.2.4 CF_CFDP_R1_SubstateRecvFileData() void CF_CFDP_R1_SubstateRecvFileData (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

Process received file data for R1.

Description

For R1, only need to digest the CRC.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

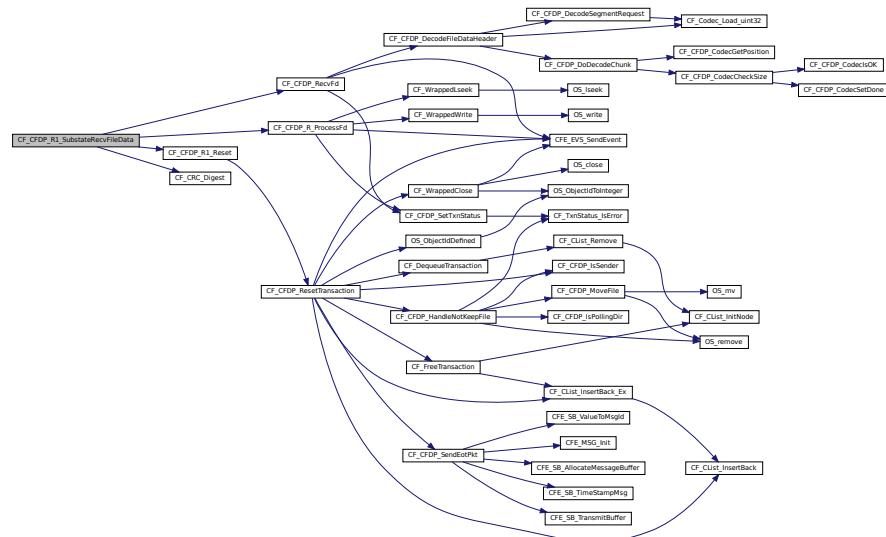
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 379 of file cf_cfdp_r.c.

References CF_CFDP_R1_Reset(), CF_CFDP_R_ProcessFd(), CF_CFDP_RecvFd(), CF_CRC_Digest(), CFE_S_UCESS, CF_Transaction::crc, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, CF_Logical_IntHeader::fd, and CF_Logical_PduBuffer::int_header.

Referenced by CF CFDP R1 Recv().

Here is the call graph for this function:



12.26.2.5 CF_CFDP_R2_CalcCrcChunk() CFE_Status_t CF_CFDP_R2_CalcCrcChunk (CFE_Transaction_t * txn)

Calculate up to the configured amount of bytes of CRC.

Description

The configuration table has a number of bytes to calculate per transaction per wakeup. At each wakeup, the file is read and this number of bytes are calculated. This function will set the checksum error condition code if the final CRC does not match.

PTFO

Increase throughput by consuming all CRC bytes per wakeup in transaction-order. This would require a change to the meaning of the value in the configuration table.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

CFE_SUCCESS	on completion.
CF_ERROR	on non-completion.

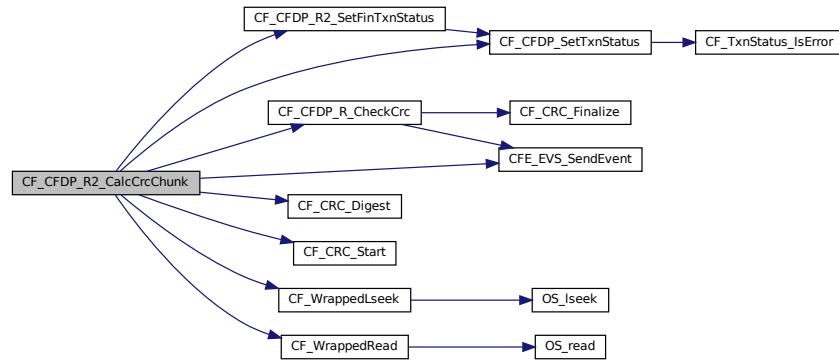
Definition at line 621 of file cf_cfdp_r.c.

References CF_RxState_Data::cached_pos, CF_AppData, CF_CFDP_FinDeliveryCode_COMPLETE, CF_CFDP_FinFileStatus_RETAINED, CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_READ_ERR_EID, CF_CFDP_R_SEEK_CRC_ERR_EID, CF_CFDP_SetTxnStatus(), CF_CRC_Digest(), CF_CRC_Start(), CF_ERROR, CF_R2_CRC_CHUNK_SIZE, CF_TxnState_R2, CF_TxnStatus_FILE_CHECKSUM_FAILURE, CF_

TxnStatus_FILE_SIZE_ERROR, CF_WrappedLseek(), CF_WrappedRead(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_StateFlags::com, CF_HkChannel_Data::counters, CF_Transaction::crc, CF_Flags_Common::crc_calc, CF_RxS2_Data::dc, CF_RxS2_Data::eof_crc, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_read, CF_HkFault::file_seek, CF_Transaction::flags, CF_RxS2_Data::fs, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Transaction::keep, OS_SEEK_SET, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_RxS2_Data::rx_crc_calc_bytes, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_R2_SubstateSendFin().

Here is the call graph for this function:



12.26.2.6 CF_CFDP_R2_Complete() void CF_CFDP_R2_Complete (CF_Transaction_t * txn,
int ok_to_send_nak)

Checks R2 transaction state for transaction completion status.

Description

This function is called anywhere there's a desire to know if the transaction has completed. It may trigger other actions by setting flags to be handled during tick processing. In order for a transaction to be complete, it must have had its meta-data PDU received, the EOF must have been received, and there must be no gaps in the file. EOF is not checked in this function, because it's only called from functions after EOF is received.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
<code>ok_to_send_nak</code>	If set to 0, suppress sending of a NAK packet

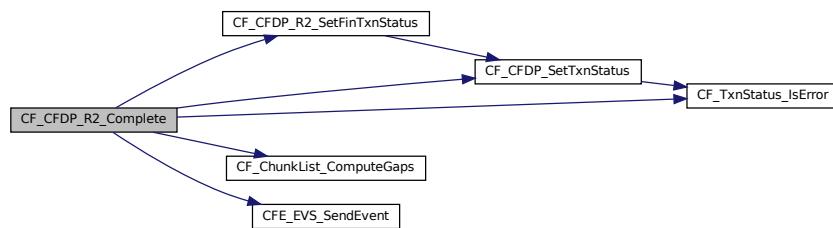
Definition at line 115 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_AppData, CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_NAK_LIMIT_ERR_EID, CF_CFDP_SetTxnStatus(), CF_ChunkList_ComputeGaps(), CF_RxSubState_FILENODATA, CF_Txn

State_R2, CF_TxnStatus_IsError(), CF_TxnStatus_NAK_LIMIT_REACHED, CF_TxnStatus_NO_ERROR, CFE_EVS_S_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_Flags_Rx::eof_recv, CF_HkCounters::fault, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, CF_ChannelConfig::nak_limit, CF_HkFault::nak_limit, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_nak, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



12.26.2.7 CF_CFDP_R2_GapCompute() void CF_CFDP_R2_GapCompute (const CF_ChunkList_t * chunks, const CF_Chunk_t * chunk, void * opaque)

Loads a single NAK segment request.

Description

This is a function callback from [CF_ChunkList_ComputeGaps\(\)](#).

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL, opaque must not be NULL.

Parameters

<code>chunks</code>	Not used, required for compatibility with CF_ChunkList_ComputeGaps
<code>chunk</code>	Pointer to a single chunk information
<code>opaque</code>	Pointer to a CF_GapComputeArgs_t object (passed via CF_ChunkList_ComputeGaps)

Definition at line 453 of file cf_cfdp_r.c.

References CF_ASSERT, CF_PDU_MAX_SEGMENTS, CF_GapComputeArgs_t::nak, CF_Logical_SegmentList::num_segments, CF_Chunk::offset, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_Logical_PduNak::scope_start, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, and CF_Chunk::size.

Referenced by CF_CFDP_R_SubstateSendNak().

```
12.26.2.8 CF_CFDP_R2_Recv() void CF_CFDP_R2_Recv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

R2 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

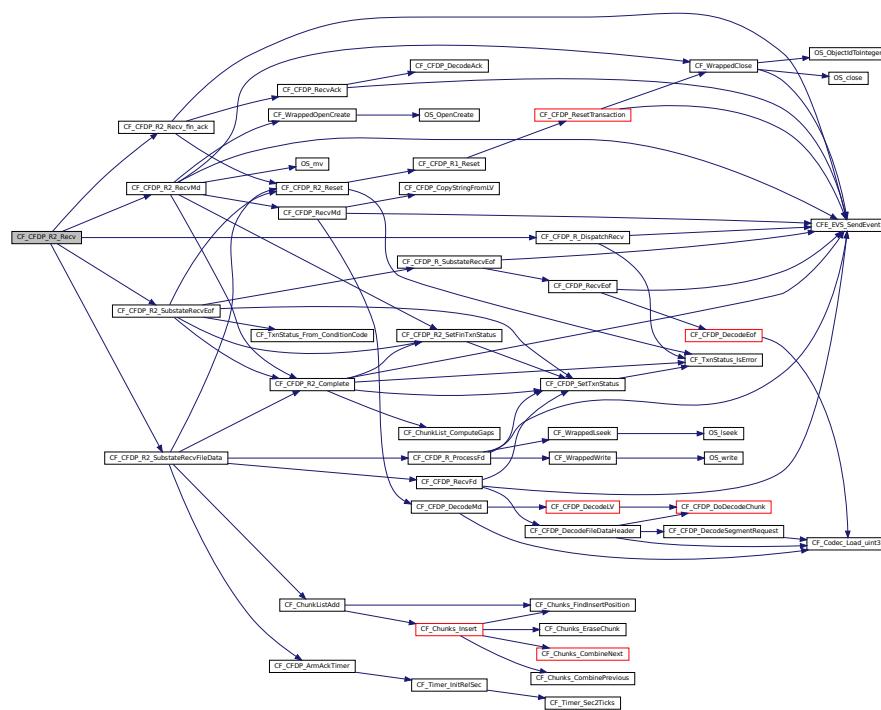
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 900 of file cf_cfdp_r.c.

References CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_EOF, CF_CFDP_FileDirective_METADATA, CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_DispatchRecv(), CF_RxSubState_EOF, CF_RxSubState_FILEDATA, CF_RxSubState_WAIT_FOR_FIN_ACK, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, and CF_CFDP_R_Substate_DispatchTable_t::state.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



```
12.26.2.9 CF_CFDP_R2_Recv_fin_ack() void CF_CFDP_R2_Recv_fin_ack (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Process receive FIN-ACK PDU.

Description

This is the end of an R2 transaction. Simply reset the transaction state.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

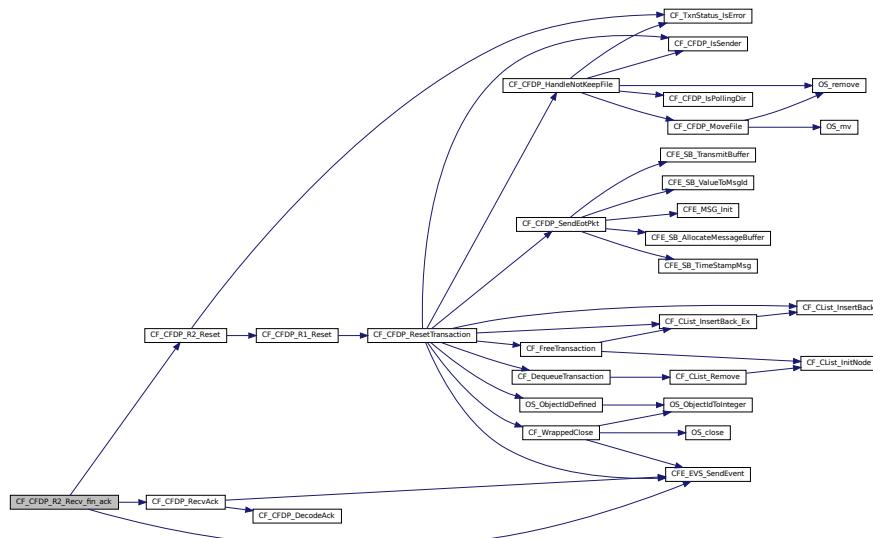
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 758 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R2_Reset(), CF_CFDP_R_PDU_FINACK_ERR_EID, CF_CFDP_RecvAck(), CF_TxnState_R2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_HkCounters::recv, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.26.2.10 CF_CFDP_R2_RecvMd() void CF_CFDP_R2_RecvMd (

```
    CF_Transaction_t * txns,
    CF_Logical_PduBuffer_t * ph )
```

Process receive metadata PDU for R2.

Description

It's possible that metadata PDU was missed in [cf_cfdp.c](#), or that it was re-sent. This function checks if it was already processed, and if not, handles it. If there was a temp file opened due to missed metadata PDU, it will move the file to the correct destination according to the metadata PDU.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

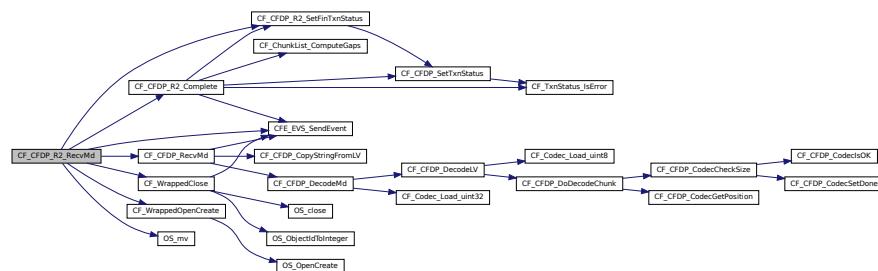
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 780 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_RxState_Data::cached_pos, CF_AppData, CF_CFDP_R2::Complete(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_EOF_MD_SIZE_ERR_EID, CF_CFDP_R_OPEN_ERR_EID, CF_CFDP_R_PDU_MD_ERR_EID, CF_CFDP_R_RENAME_ERR_EID, CF_CFDP_RecvMd(), CF_FI::LENAME_MAX_LEN, CF_PERF_ID_RENAME, CF_TxnState_R2, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_FILESTORE_REJECTION, CF_WrappedClose(), CF_WrappedOpenCreate(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_TxnFilenames::dst_filename, CF_Flags_Rx::eof_recv, CF_RxS2_Data::eof_size, CF_HkRecv::error, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_HkFault::file_rename, CF_HkFault::file_size_mismatch, CF_Transaction::flags, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, OS_FILE_FLAG_NONE, OS_mv(), OS_OBJECT_ID_UNDEFINED, OS_READ_WRITE, OS_SUCCESS, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.26.2.11 CF_CFDP_R2_Reset() void CF_CFDP_R2_Reset (CF_Transaction_t * txn)

CFDP R2 transaction reset function.

Description

Handles reset logic for R2, then calls R1 reset logic.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

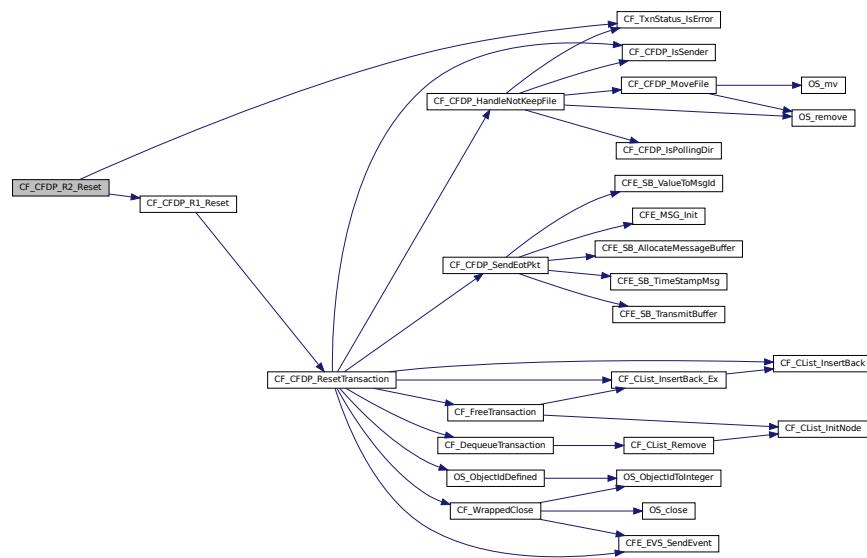
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 70 of file cf_cfdp_r.c.

References CF_Flags_Common::canceled, CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_R1_Reset(), CF_↔ RxSubState_WAIT_FOR_FIN_ACK, CF_TxnStatus_IsError(), CF_StateFlags::com, CF_RxS2_Data::eof_cc, CF_↔ Transaction::flags, CF_Transaction::history, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_↔ Flags_Rx::send_fin, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecv↔ FileData(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



12.26.2.12 CF_CFDP_R2_SetFinTxnStatus() void CF_CFDP_R2_SetFinTxnStatus (
 CF_Transaction_t * *txn*,
 CF_TxnStatus_t *txn_stat*)

Helper function to store transaction status code and set send_fin flag.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>txn_stat</i>	Status Code value to set within transaction

Definition at line 47 of file cf_cfdp_r.c.

References CF_CFDP_SetTxnStatus(), CF_Transaction::flags, CF_StateFlags::rx, and CF_Flags_Rx::send_fin.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_Init(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



12.26.2.13 CF_CFDP_R2_SubstateRecvEof() void CF_CFDP_R2_SubstateRecvEof (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

Process receive EOF for R2.

Description

For R2, need to trigger the send of EOF-ACK and then call the check complete function which will either send NAK or FIN.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

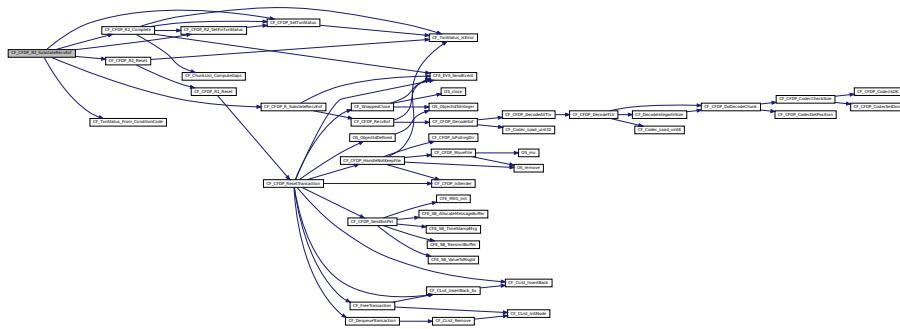
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 322 of file cf_cfdp_r.c.

References CF_Logical_PduEof::cc, CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_SetTxnStatus(), CF_F_REC_PDU_FSIZE_MISMATCH_ERROR, CF_RxSubState_FILEDATA, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_From_ConditionCode(), CFE_SUCCESS, CF_Logical_PduEof::crc, CF_Logical_IntHeader::eof, CF_RxS2_Data::eof_cc, CF_RxS2_Data::eof_crc, CF_Flags_Rx::eof_recv, CF_RxS2_Data::eof_size, CF_Transaction::flags, CF_Logical_PduBuffer::int_header, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_ack, CF_Logical_PduEof::size, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.26.2.14 CF_CFDP_R2_SubstateRecvFileData() void CF_CFDP_R2_SubstateRecvFileData (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Process received file data for R2.

Description

For R2, the CRC is checked after the whole file is received since there may be gaps. Instead, insert file received range data into chunks. Once NAK has been received, this function always checks for completion. This function also re-arms the ACK timer.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

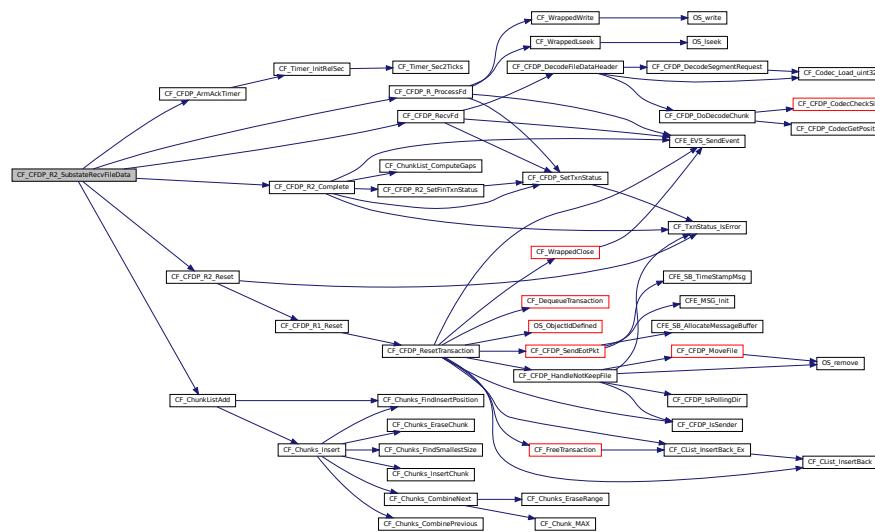
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 408 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_CFDP_ArmAckTimer(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R_ProcessFd(), CF_CFDP_RecvFd(), CF_ChunkListAdd(), CFE_SUCCESS, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_Logical_PduFileDataHeader::data_len, CF_Logical_IntHeader::fd, CF_Flags_Rx::fd_nak_sent, CF_Transaction::flags, CF_Logical_PduBuffer::int_header, CF_Logical_PduFileDataHeader::offset, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, and CF_Transaction::state_data.

Referenced by CF CFDP R2 Recv().

Here is the call graph for this function:



12.26.2.15 CF_CFDP_R2_SubstateSendFin() CFE_Status_t CF_CFDP_R2_SubstateSendFin (CF_Transaction_t * txn)

Send a FIN PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

`CFE_SUCCESS` on success. `CF_ERROR` on error.

Parameters

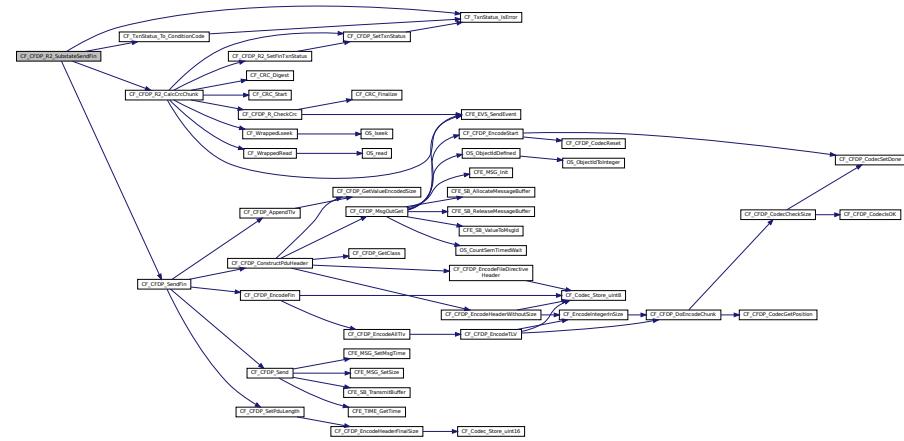
txn Pointer to the transaction object

Definition at line 721 of file cf_cfdp_r.c.

References CF_ASSERT, CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_SendFin(), CF_ERROR, CF_RxSubState_WA, IT_FOR_FIN_ACK, CF_SEND_PDU_ERROR, CF_TxnStatus_IsError(), CF_TxnStatus_To_ConditionCode(), CFE_SUCCESS, CF_StateFlags::com, CF_Flags_Common::crc_calc, CF_RxS2_Data::dc, CF_Transaction::flags, CF_RxS2_Data::fs, CF_Transaction::history, CF_RxState_Data::r2, CF_StateData::receive, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF CFDP R Tick().

Here is the call graph for this function:



12.26.2.16 CF_CFDP_R_Cancel() void CF_CFDP_R_Cancel (CF_Transaction_t * txrn)

Cancel an R transaction.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

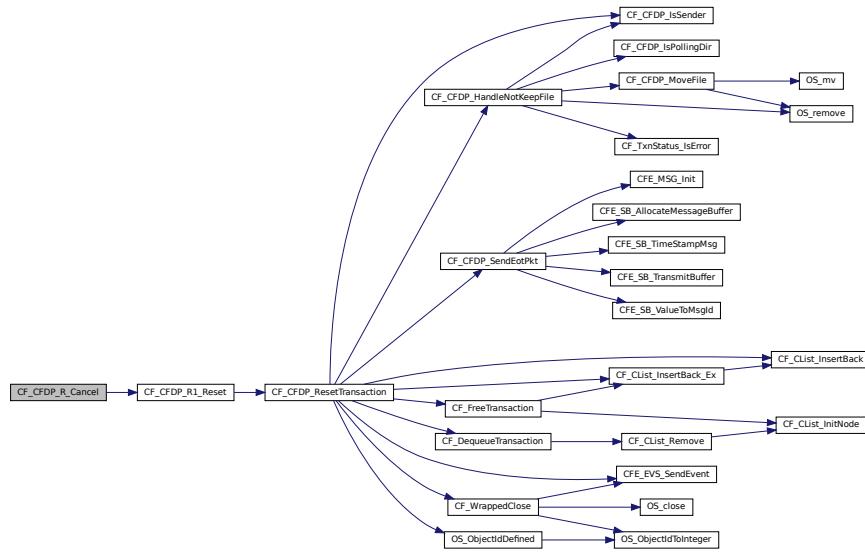
txn Pointer to the transaction object

Definition at line 926 of file cf_cfdp_r.c.

References CF_CFDP_R1_Reset(), CF_RxSubState_WAIT_FOR_FIN_ACK, CF_TxnState_R2, CF_Transaction::flags, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_fin, CF_Transaction::state, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CF_CFDP_CancelTransaction().

Here is the call graph for this function:



12.26.2.17 CF_CFDP_R_CheckCrc() `CFE_Status_t CF_CFDP_R_CheckCrc (CF_Transaction_t * txn,
 uint32 expected_crc)`

Checks that the transaction file's CRC matches expected.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

<code>CFE_SUCCESS</code>	on CRC match, otherwise <code>CF_ERROR</code> .
--------------------------	---

Parameters

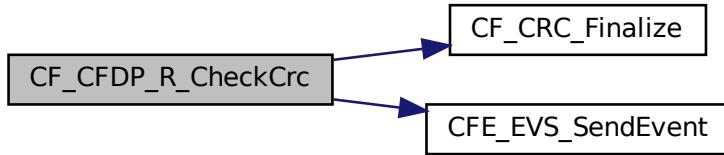
<code>txn</code>	Pointer to the transaction object
<code>expected_crc</code>	Expected CRC

Definition at line 91 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R_CRC_ERR_EID, CF_CRC_Finalize(), CF_ERROR, CF_TxnState_R2, CF_E_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket::Payload::channel_hk, CF_HkChannel_Data::counters, CF_Transaction::crc, CF_HkFault::crc_mismatch, CF_HkCounters::fault, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Crc::result, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R1_SubstateRecvEof(), and CF_CFDP_R2_CalcCrcChunk().

Here is the call graph for this function:



12.26.2.18 CF_CFDP_R_Init() void CF_CFDP_R_Init (CF_Transaction_t * txn)

Initialize a transaction structure for R.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

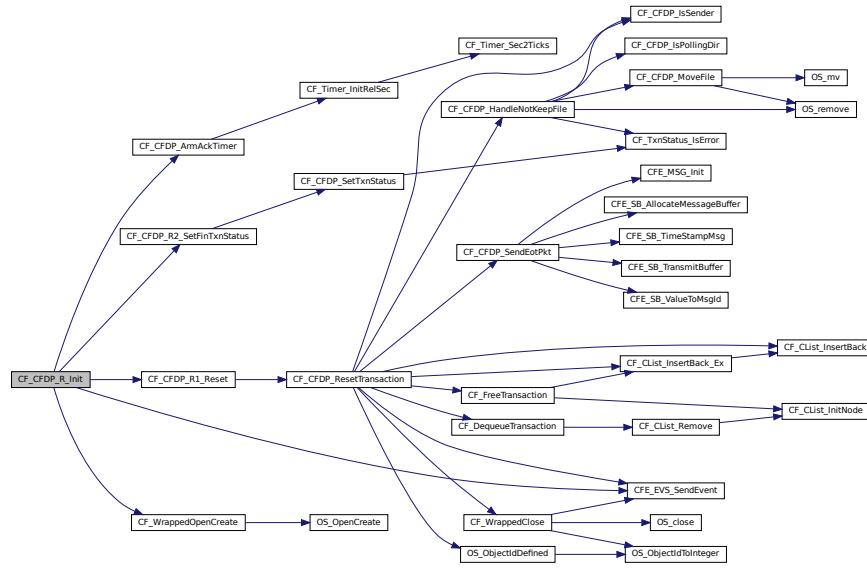
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 566 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_ArmAckTimer(), CF_CFDP_R1_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_CREAT_ERR_EID, CF_CFDP_R_TEMP_FILE_INF_EID, CF_FILENAME_MAX_PATH, CF_RxSubStateFILEDATA, CF_TxnState_R2, CF_TxnStatus_FILESTORE_REJECTION, CF_WrappedOpenCreate(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_TxnFilenames::dst_filename, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_Transaction::flags, CF_History::fnames, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, OS_FILE_FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_OBJECT_ID_UNDEFINED, OS_READ_WRITE, CF_HkPacket::Payload, CF_StateData::receive, CF_StateFlags::rx, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_ConfigTable::tmp_dir.

Referenced by CF_CFDP_RecvIdle().

Here is the call graph for this function:



12.26.2.19 CF_CFDP_R_ProcessFd() `CFE_Status_t CF_CFDP_R_ProcessFd(CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)`

Process a filedata PDU on a transaction.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

<code>CFE_SUCCESS</code>	on success. <code>CF_ERROR</code> on error.
--------------------------	---

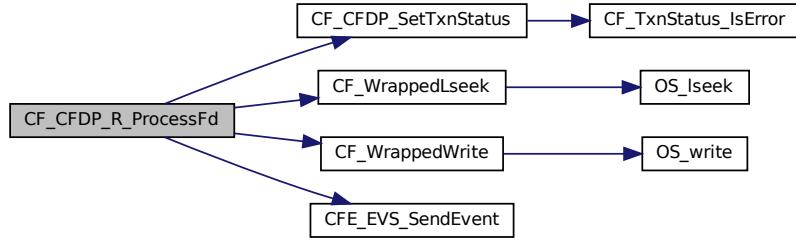
Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 190 of file cf_cfdp_r.c.

References `CF_RxState_Data::cached_pos`, `CF_AppData`, `CF_CFDP_R_SEEK_FD_ERR_EID`, `CF_CFDP_R_WRITE_ERR_EID`, `CF_CFDP_SetTxnStatus()`, `CF_ERROR`, `CF_TxnStatus_R2`, `CF_TxnStatus_FILE_SIZE_ERROR`, `CF_TxnStatus_FILESTORE_REJECTION`, `CF_WrappedLseek()`, `CF_WrappedWrite()`, `CFE_EVS_EventType_EROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_Logical_PduFileDataHeader::data_len`, `CF_Logical_PduFileDataHeader::data_ptr`, `CF_HkCounters::fault`, `CF_Logical_IntHeader::fd`, `CF_Transaction::fd`, `CF_HkRecv::file_data_bytes`, `CF_HkFault::file_seek`, `CF_HkFault::file_write`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_Logical_PduBuffer::int_header`, `CF_Logical_PduFileDataHeader::offset`, `OS_SEEK_SET`, `CF_HkPacket::Payload`, `CF_StateData::receive`, `CF_Hk`

Counters::recv, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data. Referenced by CF_CFDP_R1_SubstateRecvFileData(), and CF_CFDP_R2_SubstateRecvFileData(). Here is the call graph for this function:



12.26.2.20 CF_CFDP_R_SendInactivityEvent() `void CF_CFDP_R_SendInactivityEvent (CF_Transaction_t * txn)`

Sends an inactivity timer expired event to EVS.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 945 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R_INACT_TIMER_ERR_EID, CF_TxnState_R2, CFE_EVS_EventType_ER↔ROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel↔Data::counters, CF_HkCounters::fault, CF_Transaction::history, CF_AppData_t::hk, CF_HkFault::inactivity_timer, C↔F_HkPacket::Payload, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R_Tick().

Here is the call graph for this function:



12.26.2.21 CF_CFDP_R_SubstateRecvEof() `CFE_Status_t CF_CFDP_R_SubstateRecvEof (`

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Processing receive EOF common functionality for R1/R2.

Description

This function is used for both R1 and R2 EOF receive. It calls the unmarshaling function and then checks known transaction data against the PDU.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Return values

<code>CFE_SUCCESS</code>	on success. Returns anything else on error.
--------------------------	---

Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 250 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R_PDU_EOF_ERR_EID, CF_CFDP_R_SIZE_MISMATCH_ERR_EID, CF_CFDP_RecvEof(), CF_REC_PDU_BAD_EOF_ERROR, CF_REC_PDU_FSIZE_MISMATCH_ERROR, CF_TxnState_↔ R2, CFE_EVT_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_Hk↔ Packet_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_IntHeader::eof, CF_HkRecv::error, CF_HkCounters::fault, CF_HkFault::file_size_mismatch, CF_Transaction::flags, CF_Transaction::fsiz, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Flags_Rx::md_recv, CF_HkPacket::Payload, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_Logical_PduEof::size, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R1_SubstateRecvEof(), and CF_CFDP_R2_SubstateRecvEof().

Here is the call graph for this function:



12.26.2.22 CF_CFDP_R_SubstateSendNak() `CFE_Status_t CF_CFDP_R_SubstateSendNak (CF_Transaction_t * txn)`

Send a NAK PDU for R2.

Description

NAK PDU is sent when there are gaps in the received data. The chunks class tracks this and generates the NAK PDU by calculating gaps internally and calling `CF_CFDP_R2_GapCompute()`. There is a special case where if a metadata PDU has not been received, then a NAK packet will be sent to request another.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

CFE_SUCCESS on success. CF_ERROR on error.

Parameters

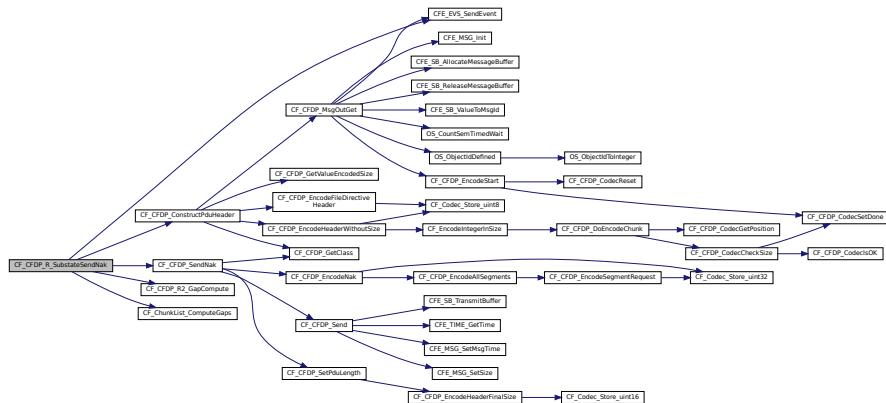
txn Pointer to the transaction object

Definition at line 485 of file cf_cfdp_r.c.

References CF_AppData, CF_Assert, CF_CFDP_ConstructPduHeader(), CF_CFDP_FileDirective_NAK, CF_CFDP_R2_GapCompute(), CF_CFDP_R_REQUEST_MD_INF_EID, CF_CFDP_SendNak(), CF_ChunkList_ComputeGaps(), CF_ERROR, CF_SEND_PDU_ERROR, CF_TxnState_R2, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_ChunkList::count, CF_HkChannel_Data::counters, CF_Flags_Rx::fd_nak_sent, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_ChunkList::max_chunks, CF_Flags_Rx::md_recv, CF_Logical_IntHeader::nak, CF_HkSent::nak_segment_requests, CF_Logical_SegmentList::num_segments, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_HkPacket::Payload, CF_History::peer_eid, CF_StateFlags::rx, CF_Logical_PduNak::scope_end, CF_Logical_PduNak::scope_start, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, CF_HkCounters::sent, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R_Tick().

Here is the call graph for this function:



```
12.26.2.23 CF_CFDP_R_Tick() void CF_CFDP_R_Tick (  
    CF_Transaction_t * txn,  
    int * cont )
```

Perform tick (time-based) processing for R transactions.

Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send ACK, NAK, and FIN. It checks for inactivity timer and processes the ACK timer. The ACK timer is what triggers re-sends of PDUs that require acknowledgment.

Assumptions, External Events, and Notes:

txn must not be NULL. cont is unused, so may be NULL

Parameters

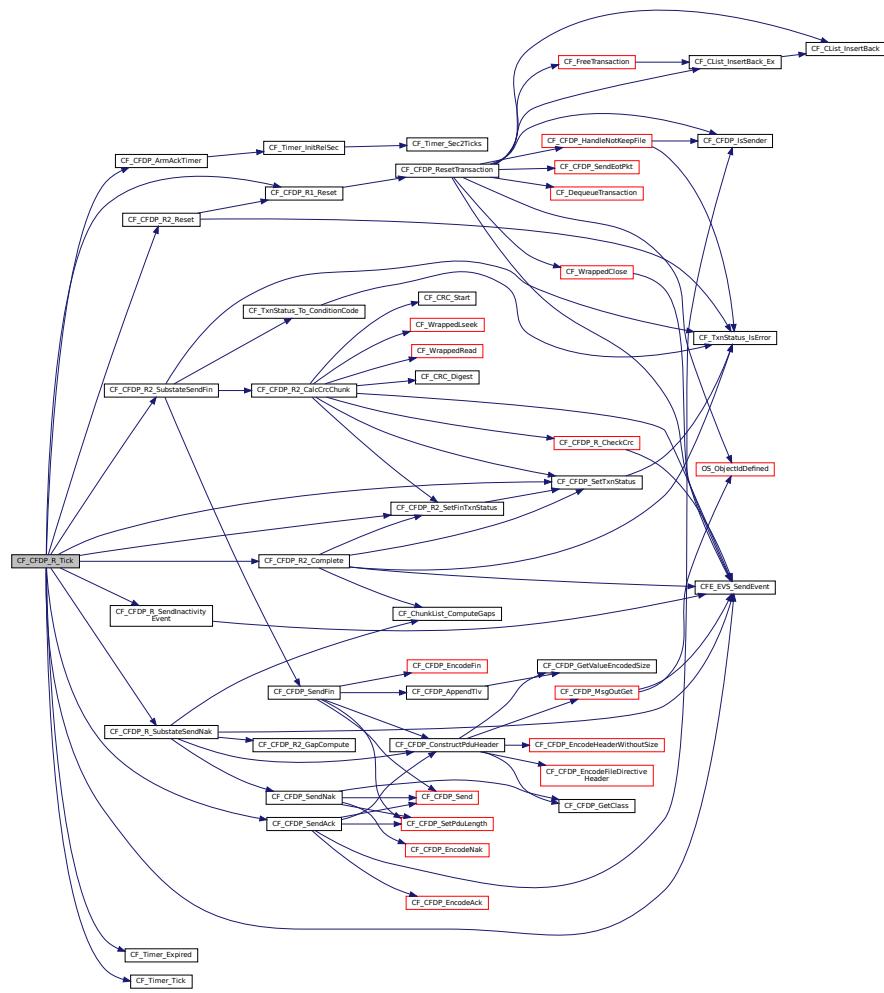
<i>txn</i>	Pointer to the transaction object
<i>cont</i>	Ignored/Unused

Definition at line 959 of file cf_cfdp_r.c.

References CF_ChannelConfig::ack_limit, CF_HkFault::ack_limit, CF_Transaction::ack_timer, CF_Flags_Common::ack_timer_armed, CF_RxS2_Data::acknak_count, CF_AppData, CF_ASSERT, CF_CFDP_AckTxnStatus_ACTIVE, CF_CFDP_ArmAckTimer(), CF_CFDP_FileDirective_EOF, CF_CFDP_R1_Reset(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_ACK_LIMIT_ERR_EID, CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_SendAck(), CF_CFDP_SetTxnStatus(), CF_RxSubState_WAIT_FOR_FIN_ACK, CF_SEND_PDU_ERROR, CF_SEND_PD_U_NO_BUF_AVAIL_ERROR, CF_Timer_Expired(), CF_Timer_Tick(), CF_TxnState_R2, CF_TxnStatus_ACK_LIMIT_NO_FIN, CF_TxnStatus_INACTIVITY_DETECTED, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_StateFlags::com, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_RxS2_Data::eof_cc, CF_HkCounters::fault, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::inactivity_fired, CF_Transaction::inactivity_timer, CF_HkPacket::Payload, CF_History::peer_eid, CF_RxState_Data::r2, CF_State_Data::receive, CF_StateFlags::rx, CF_Flags_Rx::send_ack, CF_Flags_Rx::send_fin, CF_Flags_Rx::send_nak, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CF_CFDP_TickTransactions().

Here is the call graph for this function:



12.27 apps/cf/fsw/src/cf_cfdp_r.h File Reference

```
#include "cf_cfdp.h"
```

Data Structures

- struct `CF_GapComputeArgs_t`

Argument for Gap Compute function.

Functions

- void **CF_CFDP_R1_Recv** (**CF_Transaction_t** *txn, **CF_Logical_PduBuffer_t** *ph)
R1 receive PDU processing.
 - void **CF_CFDP_R2_Recv** (**CF_Transaction_t** *txn, **CF_Logical_PduBuffer_t** *ph)
R2 receive PDU processing.
 - void **CF_CFDP_R_Tick** (**CF_Transaction_t** *txn, int *cont)

Perform tick (time-based) processing for R transactions.

- void `CF_CFDP_R_Cancel` (`CF_Transaction_t` *`txn`)
Cancel an R transaction.
- void `CF_CFDP_R_Init` (`CF_Transaction_t` *`txn`)
Initialize a transaction structure for R.
- void `CF_CFDP_R2_SetFinTxnStatus` (`CF_Transaction_t` *`txn`, `CF_TxnStatus_t` `txn_stat`)
Helper function to store transaction status code and set send_fin flag.
- void `CF_CFDP_R1_Reset` (`CF_Transaction_t` *`txn`)
CFDP R1 transaction reset function.
- void `CF_CFDP_R2_Reset` (`CF_Transaction_t` *`txn`)
CFDP R2 transaction reset function.
- `CFE_Status_t CF_CFDP_R_CheckCrc` (`CF_Transaction_t` *`txn`, `uint32` `expected_crc`)
Checks that the transaction file's CRC matches expected.
- void `CF_CFDP_R2_Complete` (`CF_Transaction_t` *`txn`, int `ok_to_send_nak`)
Checks R2 transaction state for transaction completion status.
- `CFE_Status_t CF_CFDP_R_ProcessFd` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process a filedata PDU on a transaction.
- `CFE_Status_t CF_CFDP_R_SubstateRecvEof` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Processing receive EOF common functionality for R1/R2.
- void `CF_CFDP_R1_SubstateRecvEof` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process receive EOF for R1.
- void `CF_CFDP_R2_SubstateRecvEof` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process receive EOF for R2.
- void `CF_CFDP_R1_SubstateRecvFileData` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process received file data for R1.
- void `CF_CFDP_R2_SubstateRecvFileData` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process received file data for R2.
- void `CF_CFDP_R2_GapCompute` (`const CF_ChunkList_t` *`chunks`, `const CF_Chunk_t` *`chunk`, `void` *`opaque`)
Loads a single NAK segment request.
- `CFE_Status_t CF_CFDP_R_SubstateSendNak` (`CF_Transaction_t` *`txn`)
Send a NAK PDU for R2.
- `CFE_Status_t CF_CFDP_R2_CalcCrcChunk` (`CF_Transaction_t` *`txn`)
Calculate up to the configured amount of bytes of CRC.
- `CFE_Status_t CF_CFDP_R2_SubstateSendFin` (`CF_Transaction_t` *`txn`)
Send a FIN PDU.
- void `CF_CFDP_R2_Recv_fin_ack` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process receive FIN-ACK PDU.
- void `CF_CFDP_R2_RecvMd` (`CF_Transaction_t` *`txn`, `CF_Logical_PduBuffer_t` *`ph`)
Process receive metadata PDU for R2.
- void `CF_CFDP_R_SendInactivityEvent` (`CF_Transaction_t` *`txn`)
Sends an inactivity timer expired event to EVS.

12.27.1 Detailed Description

Implementation related to CFDP Receive File transactions

This file contains various state handling routines for transactions which are receiving a file.

12.27.2 Function Documentation

12.27.2.1 CF_CFDP_R1_Recv() void CF_CFDP_R1_Recv (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

R1 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

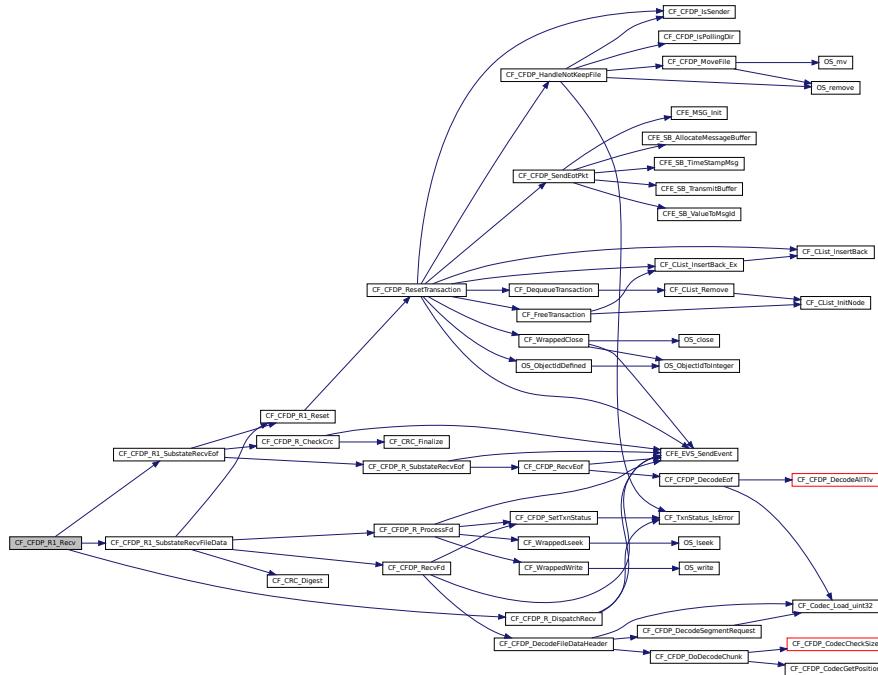
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 882 of file cf_cfdp_r.c.

References CF_CFDP_FileDirective_EOF, CF_CFDP_R1_SubstateRecvEof(), CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R_DispatchRecv(), CF_RxSubState_EOF, CF_RxSubState_FILEDATA, CF_RxSubState_WAIT_FOR_FIN_ACK, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, and CF_CFDP_R_SubstateDispatchTable_t::state.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.27.2.2 CF_CFDP_R1_Reset() void CF_CFDP_R1_Reset (

```
CF_Transaction_t * txn )
```

CFDP R1 transaction reset function.

Description

All R transactions use this call to indicate the transaction state can be returned to the system. While this function currently only calls [CF_CFDP_ResetTransaction\(\)](#), it is here as a placeholder.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

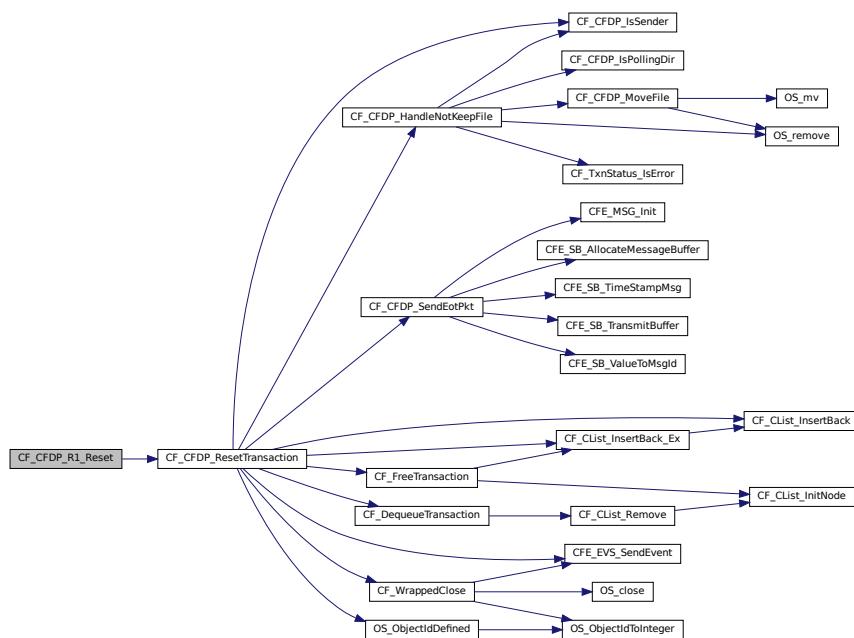
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 59 of file cf_cfdp_r.c.

References [CF_CFDP_ResetTransaction\(\)](#).

Referenced by [CF_CFDP_R1_SubstateRecvEof\(\)](#), [CF_CFDP_R1_SubstateRecvFileData\(\)](#), [CF_CFDP_R2_Reset\(\)](#), [CF_CFDP_R_Cancel\(\)](#), [CF_CFDP_R_Init\(\)](#), and [CF_CFDP_R_Tick\(\)](#).

Here is the call graph for this function:



12.27.2.3 CF_CFDP_R1_SubstateRecvEof() void CF_CFDP_R1_SubstateRecvEof (
`CF_Transaction_t * txn,`
`CF_Logical_PduBuffer_t * ph`)

Process receive EOF for R1.

Description

Only need to confirm CRC for R1.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

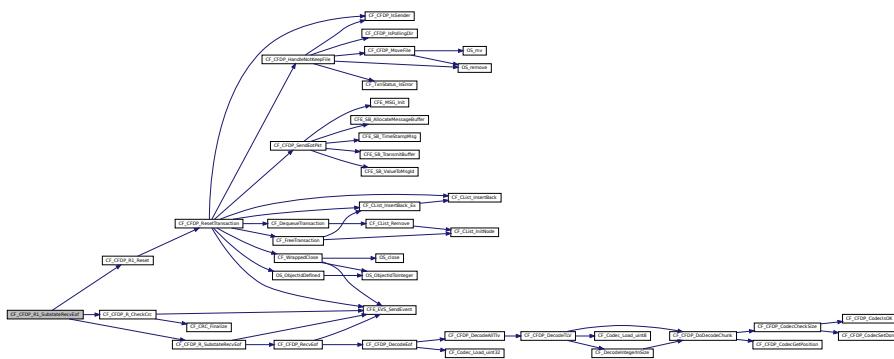
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 290 of file cf_cfdp_r.c.

References CF_CFDP_R1_Reset(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_SubstateRecvEof(), CFE_SUCCESS, CF_Logical_PduEof::crc, CF_Logical_IntHeader::eof, CF_Logical_PduBuffer::int_header, and CF_Transaction::keep.

Referenced by CF_CFDP_R1_Recv().

Here is the call graph for this function:



12.27.2.4 CF_CFDP_R1_SubstateRecvFileData()

```
void CF_CFDP_R1_SubstateRecvFileData (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Process received file data for R1.

Description

For R1, only need to digest the CRC.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

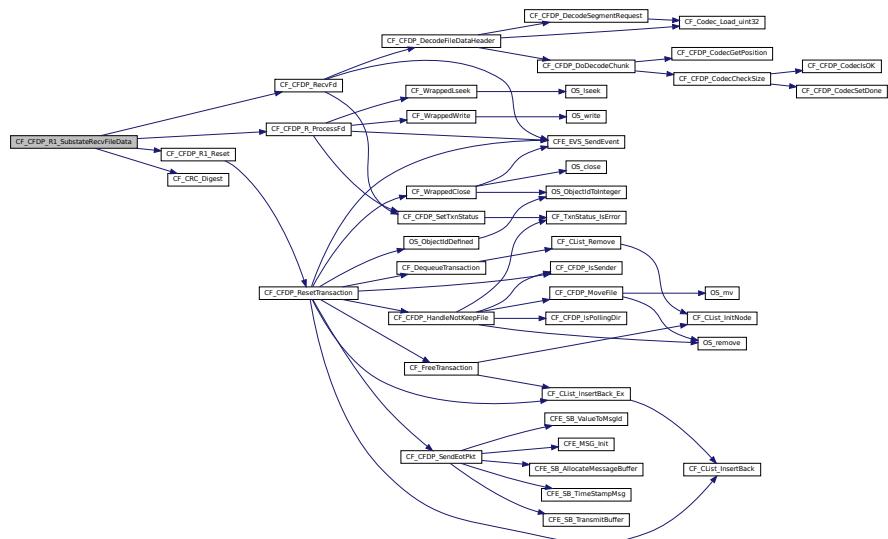
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 379 of file cf_cfdp_r.c.

References CF_CFDP_R1_Reset(), CF_CFDP_R_ProcessFd(), CF_CFDP_RecvFd(), CF_CRC_Digest(), CFE_S↔UCCES, CF_Transaction::crc, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, CF_Logical_IntHeader::fd, and CF_Logical_PduBuffer::int_header.

Referenced by CF_CFDP_R1_Recv().

Here is the call graph for this function:



12.27.2.5 CF_CFDP_R2_CalcCrcChunk() CFE_Status_t CF_CFDP_R2_CalcCrcChunk (CFE_Transaction_t * txn)

Calculate up to the configured amount of bytes of CRC.

Description

The configuration table has a number of bytes to calculate per transaction per wakeup. At each wakeup, the file is read and this number of bytes are calculated. This function will set the checksum error condition code if the final CRC does not match.

PTFO

Increase throughput by consuming all CRC bytes per wakeup in transaction-order. This would require a change to the meaning of the value in the configuration table.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

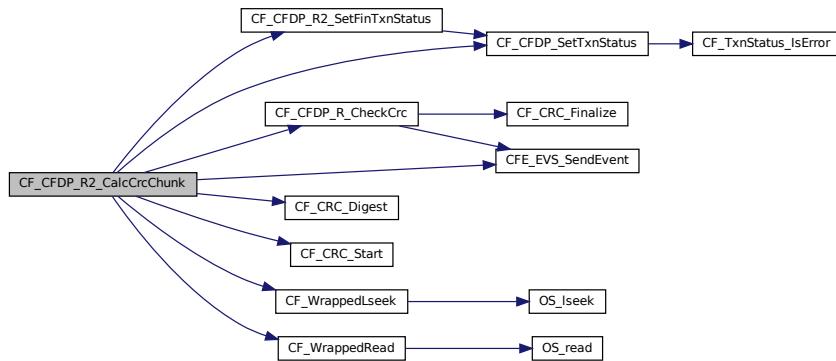
<code>CFE_SUCCESS</code>	on completion.
<code>CF_ERROR</code>	on non-completion.

Definition at line 621 of file cf_cfdp_r.c.

References CF_RxState_Data::cached_pos, CF_AppData, CF_CFDP_FinDeliveryCode_COMPLETE, CF_CFDP_FinFileStatus_RETAINED, CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_CheckCrc(), CF_CFDP_R_READ_ERR_EID, CF_CFDP_R_SEEK_CRC_ERR_EID, CF_CFDP_SetTxnStatus(), CF_CRC_Digest(), CF_CRC_Start(), CF_ERROR, CF_R2_CRC_CHUNK_SIZE, CF_TxnState_R2, CF_TxnStatus_FILE_CHECKSUM_FAILURE, CF_TxnStatus_FILE_SIZE_ERROR, CF_WrappedLseek(), CF_WrappedRead(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_StateFlags::com, CF_HkChannel_Data::counters, CF_Transaction::crc, CF_Flags_Common::crc_calc, CF_RxS2_Data::dc, CF_RxS2_Data::eof_crc, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_read, CF_HkFault::file_seek, CF_Transaction::flags, CF_RxS2_Data::fs, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Transaction::keep, OS_SEEK_SET, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_RxS2_Data::rx_crc_calc_bytes, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_R2_SubstateSendFin().

Here is the call graph for this function:



12.27.2.6 CF_CFDP_R2_Complete() void CF_CFDP_R2_Complete (
`CF_Transaction_t * txn,`
`int ok_to_send_nak)`

Checks R2 transaction state for transaction completion status.

Description

This function is called anywhere there's a desire to know if the transaction has completed. It may trigger other actions by setting flags to be handled during tick processing. In order for a transaction to be complete, it must have had its meta-data PDU received, the EOF must have been received, and there must be no gaps in the file. EOF is not checked in this function, because it's only called from functions after EOF is received.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

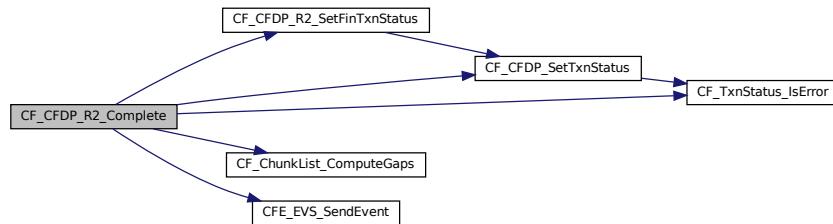
<code>txn</code>	Pointer to the transaction object
<code>ok_to_send_nak</code>	If set to 0, suppress sending of a NAK packet

Definition at line 115 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_AppData, CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_NAK_LIMIT_ERR_EID, CF_CFDP_SetTxnStatus(), CF_ChunkList_ComputeGaps(), CF_RxSubState_FILEDATA, CF_TxnState_R2, CF_TxnStatus_IsError(), CF_TxnStatus_NAK_LIMIT_REACHED, CF_TxnStatus_NO_ERROR, CFE_EVT_S_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_Flags_Rx::eof_recv, CF_HkCounters::fault, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, CF_ChannelConfig::nak_limit, CF_HkFault::nak_limit, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_nak, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



12.27.2.7 CF_CFDP_R2_GapCompute() `void CF_CFDP_R2_GapCompute (`
`const CF_ChunkList_t * chunks,`
`const CF_Chunk_t * chunk,`
`void * opaque)`

Loads a single NAK segment request.

Description

This is a function callback from [CF_ChunkList_ComputeGaps\(\)](#).

Assumptions, External Events, and Notes:

`chunks` must not be NULL, `chunk` must not be NULL, `opaque` must not be NULL.

Parameters

<code>chunks</code>	Not used, required for compatibility with <code>CF_ChunkList_ComputeGaps</code>
<code>chunk</code>	Pointer to a single chunk information
<code>opaque</code>	Pointer to a CF_GapComputeArgs_t object (passed via <code>CF_ChunkList_ComputeGaps</code>)

Definition at line 453 of file cf_cfdp_r.c.

References CF_ASSERT, CF_PDU_MAX_SEGMENTS, CF_GapComputeArgs_t::nak, CF_Logical_SegmentList::num_segments, CF_Chunk::offset, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_Logical_PduNak::scope_start, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, and CF

_Chunk::size.
Referenced by CF_CFDP_R_SubstateSendNak().

12.27.2.8 CF_CFDP_R2_Recv() void CF_CFDP_R2_Recv (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

R2 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

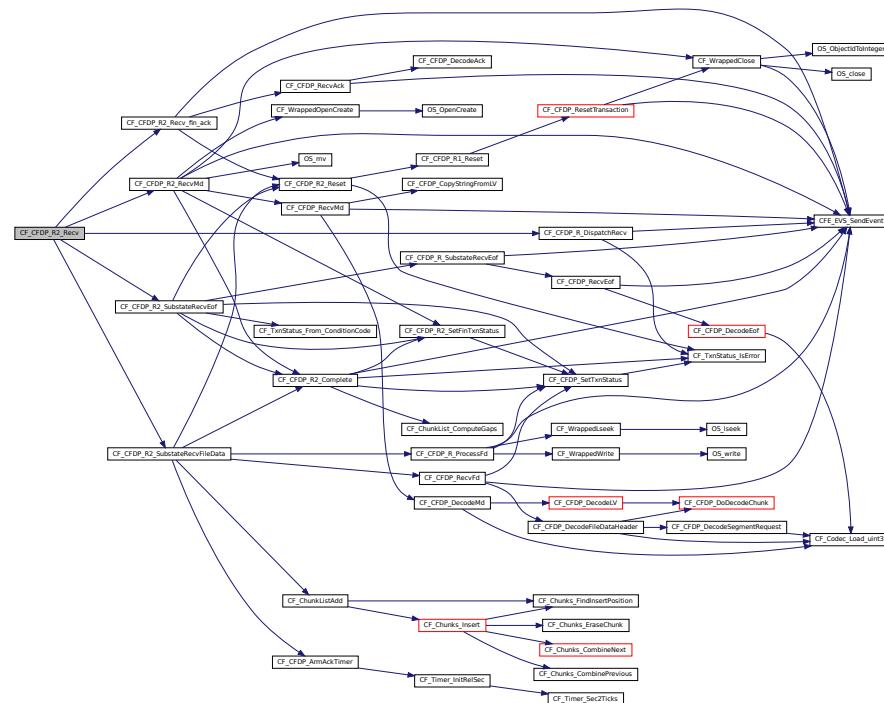
txn	Pointer to the transaction object
ph	Pointer to the PDU information

Definition at line 900 of file cf_cfdp_r.c.

References CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_EOF, CF_CFDP_FileDirective_METADATA, CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecvFileData(), CF_CFDP_R_DispatchRecv(), CF_RxSubState_EOF, CF_RxSubState_FILEDATA, CF_RxSubState_WAIT_FOR_FIN_ACK, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, and CF_CFDP_R_Substate_DispatchTable_t::state.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



```
12.27.2.9 CF_CFDP_R2_Recv_fin_ack() void CF_CFDP_R2_Recv_fin_ack (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Process receive FIN-ACK PDU.

Description

This is the end of an R2 transaction. Simply reset the transaction state.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

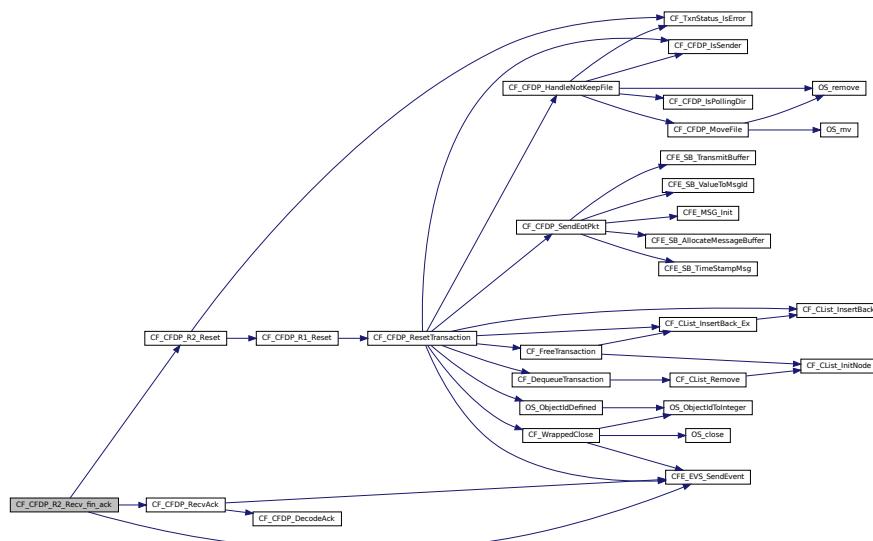
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 758 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R2_Reset(), CF_CFDP_R_PDU_FINACK_ERR_EID, CF_CFDP_RecvAck(), CF_TxnState_R2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_Hk↔Packet_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::history, CF_App↔Data_t::hk, CF_HkPacket::Payload, CF_HkCounters::recv, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



```
12.27.2.10 CF_CFDP_R2_RecvMd() void CF_CFDP_R2_RecvMd (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Process receive metadata PDU for R2.

Description

It's possible that metadata PDU was missed in [cf_cfdp.c](#), or that it was re-sent. This function checks if it was already processed, and if not, handles it. If there was a temp file opened due to missed metadata PDU, it will move the file to the correct destination according to the metadata PDU.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

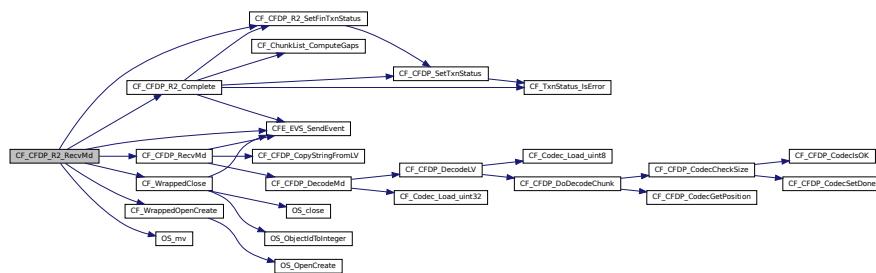
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 780 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_RxState_Data::cached_pos, CF_AppData, CF_CFDP_R2::Complete(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_EOF_MD_SIZE_ERR_EID, CF_CFDP_R_OPEN_ERR_EID, CF_CFDP_R_PDU_MD_ERR_EID, CF_CFDP_R_RENAME_ERR_EID, CF_CFDP_RecvMd(), CF_FI::LENAME_MAX_LEN, CF_PERF_ID_RENAME, CF_TxnState_R2, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_FILESTORE_REJECTION, CF_WrappedClose(), CF_WrappedOpenCreate(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_TxnFilenames::dst_filename, CF_Flags_Rx::eof_recv, CF_RxS2_Data::eof_size, CF_HkRecv::error, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_HkFault::file_rename, CF_HkFault::file_size_mismatch, CF_Transaction::flags, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, OS_FILE_FLAG_NONE, OS_mv(), OS_OBJECT_ID_UNDEFINED, OS_READ_WRITE, OS_SUCCESS, CF_HkPacket::Payload, CF_RxState_Data::r2, CF_StateData::receive, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.27.2.11 CF_CFDP_R2_Reset() void CF_CFDP_R2_Reset (CF_Transaction_t * txn)

CFDP R2 transaction reset function.

Description

Handles reset logic for R2, then calls R1 reset logic.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

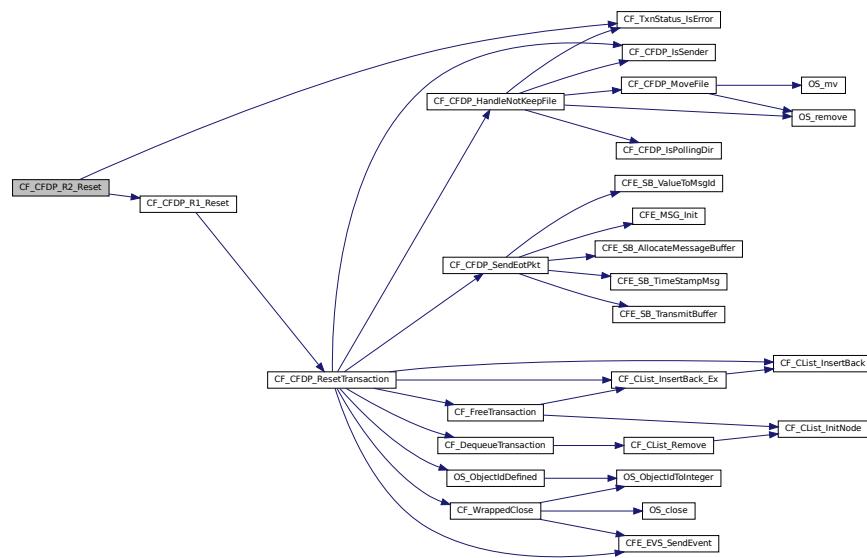
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 70 of file cf_cfdp_r.c.

References CF_Flags_Common::canceled, CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_R1_Reset(), CF_↔ RxSubState_WAIT_FOR_FIN_ACK, CF_TxnStatus_IsError(), CF_StateFlags::com, CF_RxS2_Data::eof_cc, CF_↔ Transaction::flags, CF_Transaction::history, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_↔ Flags_Rx::send_fin, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_R2_Recv_fin_ack(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R2_SubstateRecv↔ FileData(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



```

12.27.2.12 CF_CFDP_R2_SetFinTxnStatus() void CF_CFDP_R2_SetFinTxnStatus (
    CF_Transaction_t * txn,
    CF_TxnStatus_t txn_stat )
  
```

Helper function to store transaction status code and set send_fin flag.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>txn_stat</i>	Status Code value to set within transaction

Definition at line 47 of file cf_cfdp_r.c.

References CF_CFDP_SetTxnStatus(), CF_Transaction::flags, CF_StateFlags::rx, and CF_Flags_Rx::send_fin.

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R2_Complete(), CF_CFDP_R2_RecvMd(), CF_CFDP_R2_SubstateRecvEof(), CF_CFDP_R_Init(), and CF_CFDP_R_Tick().

Here is the call graph for this function:



12.27.2.13 CF_CFDP_R2_SubstateRecvEof() void CF_CFDP_R2_SubstateRecvEof (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

Process receive EOF for R2.

Description

For R2, need to trigger the send of EOF-ACK and then call the check complete function which will either send NAK or FIN.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

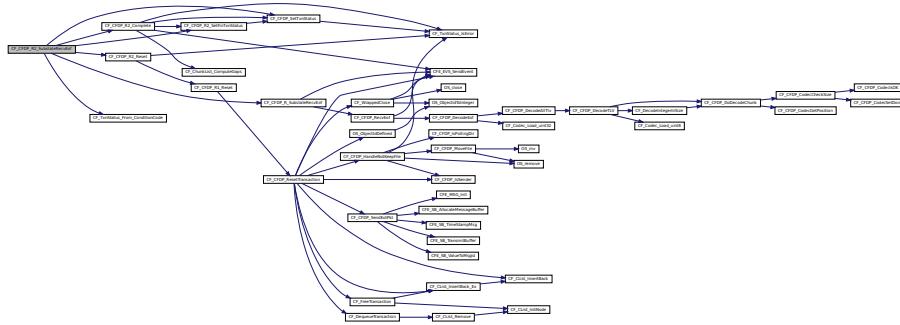
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 322 of file cf_cfdp_r.c.

References CF_Logical_PduEof::cc, CF_CFDP_ConditionCode_NO_ERROR, CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_SubstateRecvEof(), CF_CFDP_SetTxnStatus(), CF_F_REC_PDU_FSIZE_MISMATCH_ERROR, CF_RxSubState_FILEDATA, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_From_ConditionCode(), CFE_SUCCESS, CF_Logical_PduEof::crc, CF_Logical_IntHeader::eof, CF_RxS2_Data::eof_cc, CF_RxS2_Data::eof_crc, CF_Flags_Rx::eof_recv, CF_RxS2_Data::eof_size, CF_Transaction::flags, CF_Logical_PduBuffer::int_header, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_ack, CF_Logical_PduEof::size, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.27.2.14 CF_CFDP_R2_SubstateRecvFileData() void CF_CFDP_R2_SubstateRecvFileData (CF_Transaction_t * *txn*, CF_Logical_PduBuffer_t * *ph*)

Process received file data for R2.

Description

For R2, the CRC is checked after the whole file is received since there may be gaps. Instead, insert file received range data into chunks. Once NAK has been received, this function always checks for completion. This function also re-arms the ACK timer.

Assumptions, External Events, and Notes:

txn must not be NULL. *ph* must not be NULL.

Parameters

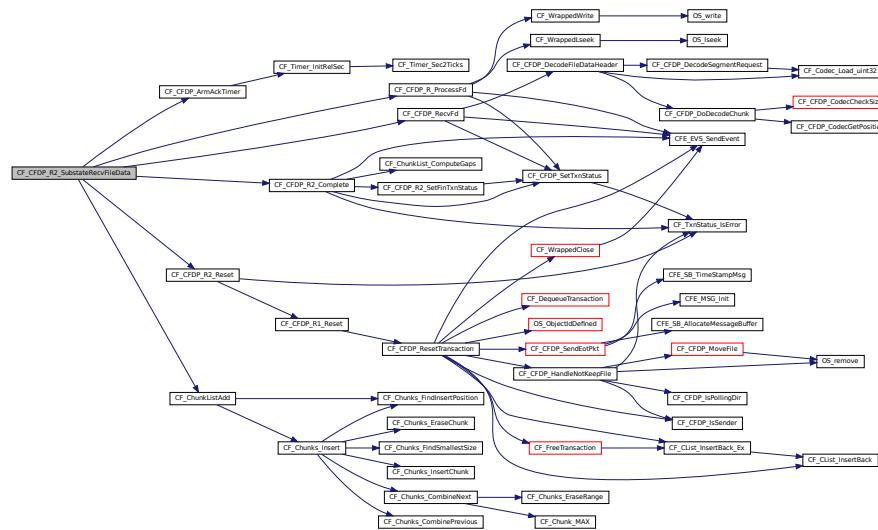
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 408 of file cf_cfdp_r.c.

References CF_RxS2_Data::acknak_count, CF_CFDP_ArmAckTimer(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R_ProcessFd(), CF_CFDP_RecvFd(), CF_ChunkListAdd(), CFE_SUCCESS, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_Logical_PduFileHeader::data_len, CF_Logical_InfHeader::fd, CF_Flags_Rx::fd_nak_sent, CF_Transaction::flags, CF_Logical_PduBuffer::int_header, CF_Logical_PduFileHeader::offset, CF_RxState_Data::r2, CF_StateData::receive, CF_StateFlags::rx, and CF_Transaction::state_data.

Referenced by CF_CFDP_R2_Recv().

Here is the call graph for this function:



12.27.2.15 CF_CFDP_R2_SubstateSendFin() CFE_Status_t CF_CFDP_R2_SubstateSendFin (

```
CF_Transaction_t * txn )
```

Send a FIN PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

`CFE_SUCCESS` on success. `CF_ERROR` on error.

Parameters

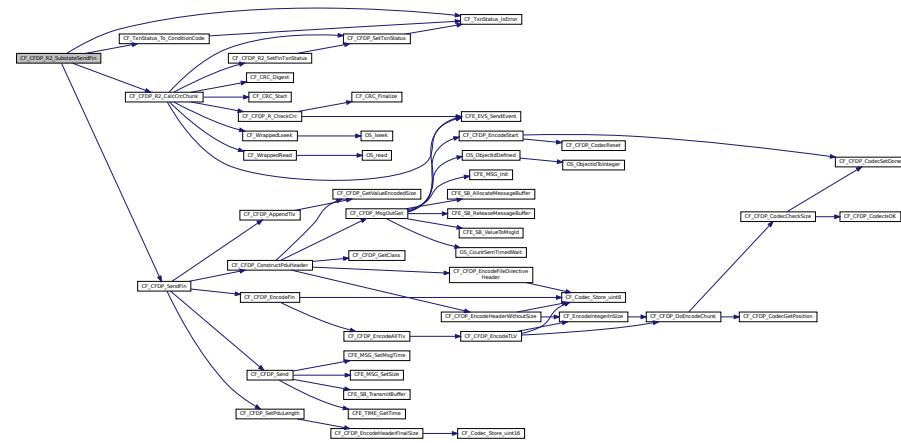
txn Pointer to the transaction object

Definition at line 721 of file cf_cfdp_r.c.

References CF Assert, CF CFDP R2 CalcCrcChunk(), CF CFDP SendFin(), CF ERROR, CF RxSubState_WA, IT_FOR_FIN_ACK, CF SEND PDU_ERROR, CF TxnStatus_IsError(), CF TxnStatus_To_ConditionCode(), CFE SUCCESS, CF StateFlags::com, CF Flags_Common::crc_calc, CF RxS2_Data::dc, CF Transaction::flags, CF RxS2_Data::fs, CF Transaction::history, CF RxState_Data::r2, CF StateData::receive, CF Transaction::state_data, CF RxState Data::sub state, and CF History::txn stat.

Referenced by CF CFDP R Tick().

Here is the call graph for this function:



12.27.2.16 CF_CFDP_R_Cancel() void CF_CFDP_R_Cancel (CF_Transaction_t * txrn)

Cancel an R transaction.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

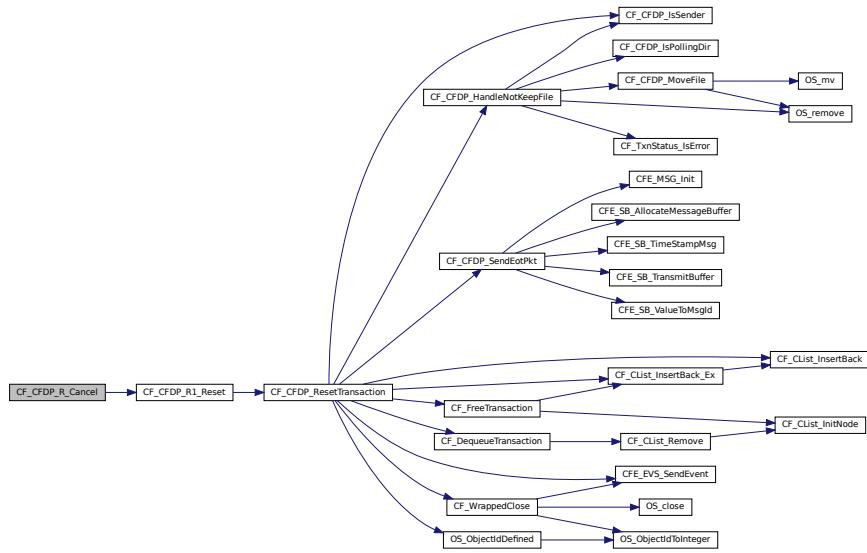
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 926 of file cf_cfdp_r.c.

References CF_CFDP_R1_Reset(), CF_RxSubState_WAIT_FOR_FIN_ACK, CF_TxnState_R2, CF_Transaction::flags, CF_StateData::receive, CF_StateFlags::rx, CF_Flags_Rx::send_fin, CF_Transaction::state, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CE_CEDP_CancelTransaction().

Here is the call graph for this function:



12.27.2.17 CF_CFDP_R_CheckCrc() `CFE_Status_t CF_CFDP_R_CheckCrc (`
`CF_Transaction_t * txn,`
`uint32 expected_crc)`

Checks that the transaction file's CRC matches expected.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

<code>CFE_SUCCESS</code>	on CRC match, otherwise <code>CF_ERROR</code> .
--------------------------	---

Parameters

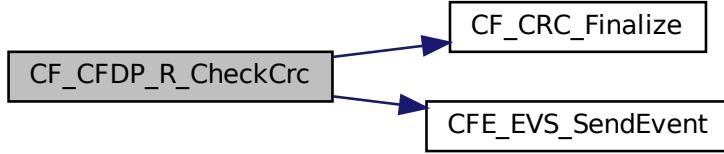
<code>txn</code>	Pointer to the transaction object
<code>expected_crc</code>	Expected CRC

Definition at line 91 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_R_CRC_ERR_EID, CF_CRC_Finalize(), CF_ERROR, CF_TxnState_R2, CF_E_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket::Payload::channel_hk, CF_HkChannel_Data::counters, CF_Transaction::crc, CF_HkFault::crc_mismatch, CF_HkCounters::fault, CF_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Crc::result, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CF_CFDP_R1_SubstateRecvEof(), and CF_CFDP_R2_CalcCrcChunk().

Here is the call graph for this function:



12.27.2.18 CF_CFDP_R_Init() void CF_CFDP_R_Init (CF_Transaction_t * txn)

Initialize a transaction structure for R.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

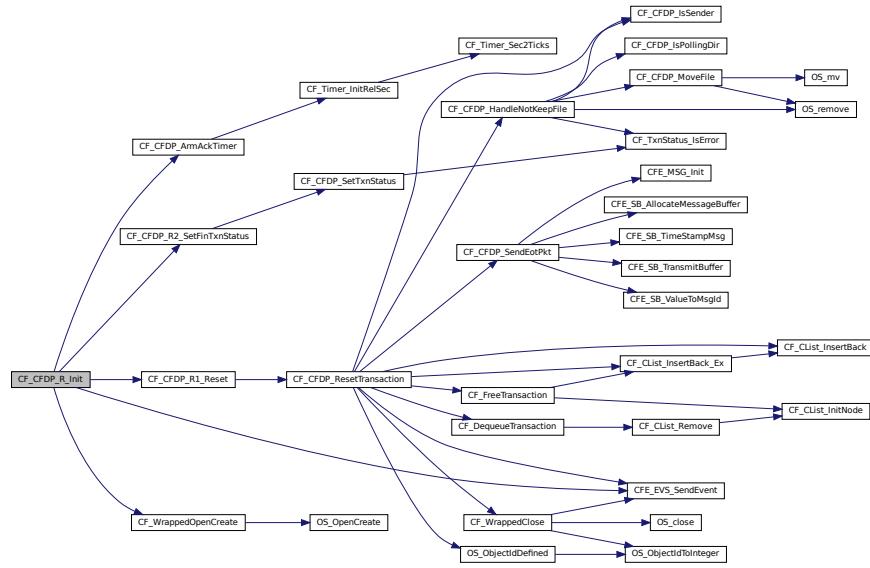
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 566 of file cf_cfdp_r.c.

References CF_AppData, CF_CFDP_ArmAckTimer(), CF_CFDP_R1_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R_CREAT_ERR_EID, CF_CFDP_R_TEMP_FILE_INF_EID, CF_FILENAME_MAX_PATH, CF_RxSubStateFILEDATA, CF_TxnState_R2, CF_TxnStatus_FILESTORE_REJECTION, CF_WrappedOpenCreate(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_TxnFilenames::dst_filename, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_Transaction::flags, CF_History::fnames, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::md_recv, OS_FILE_FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_OBJECT_ID_UNDEFINED, OS_READ_WRITE, CF_HkPacket::Payload, CF_StateData::receive, CF_StateFlags::rx, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_RxState_Data::sub_state, and CF_ConfigTable::tmp_dir.

Referenced by CF_CFDP_RecvIdle().

Here is the call graph for this function:



```
12.27.2.19 CF_CFDP_R_ProcessFd() CFE_Status_t CF_CFDP_R_ProcessFd (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

Process a filedata PDU on a transaction.

Assumptions, External Events, and Notes:

txn must not be `NULL`.

Return values

CFE_SUCCESS on success. *CF_ERROR* on error.

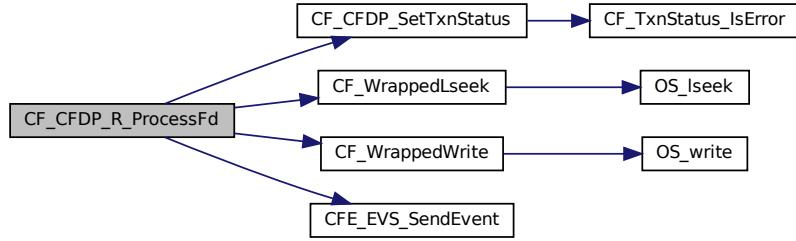
Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 190 of file cf_cfdp_r.c.

References CF_RxState_Data::cached_pos, CF_AppData, CF_CFDP_R_SEEK_FD_ERR_EID, CF_CFDP_R_WRITE_ERR_EID, CF_CFDP_SetTxnStatus(), CF_ERROR, CF_TxnState_R2, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_FILESTORE_REJECTION, CF_WrappedLseek(), CF_WrappedWrite(), CFE_EVS_EventType_EROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, CF_HkCounters::fault, CF_Logical_IntHeader::fd, CF_Transaction::fd, CF_HkRecv::file_data_bytes, CF_HkFault::file_seek, CF_HkFault::file_write, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Logical_PduFileDataHeader::offset, OS_SEEK_SET, CF_HkPacket::Payload, CF_StateData::receive, CF_Hk

Counters::recv, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.
Referenced by CF_CFDP_R1_SubstateRecvFileData(), and CF_CFDP_R2_SubstateRecvFileData().
Here is the call graph for this function:



12.27.2.20 CF_CFDP_R_SendInactivityEvent()

```
void CF_CFDP_R_SendInactivityEvent (
    CF_Transaction_t * txn )
```

Sends an inactivity timer expired event to EVS.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 945 of file cf_cfdp_r.c.

References `CF_AppData`, `CF_CFDP_R_INACT_TIMER_ERR_EID`, `CF_TxnState_R2`, `CFE_EVS_EventType_ER` ↔ `ROR`, `CFE_EVS_SendEvent()`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel::Data::counters`, `CF_HkCounters::fault`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_HkFault::inactivity_timer`, `C_F_HkPacket::Payload`, `CF_History::seq_num`, `CF_History::src_eid`, and `CF_Transaction::state`.

Referenced by `CF_CFDP_R_Tick()`.

Here is the call graph for this function:



12.27.2.21 CF_CFDP_R_SubstateRecvEof()

```
CFE_Status_t CF_CFDP_R_SubstateRecvEof (
```

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Processing receive EOF common functionality for R1/R2.

Description

This function is used for both R1 and R2 EOF receive. It calls the unmarshaling function and then checks known transaction data against the PDU.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Return values

<code>CFE_SUCCESS</code>	on success. Returns anything else on error.
--------------------------	---

Parameters

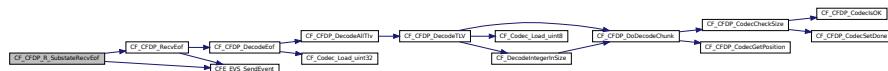
<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 250 of file cf_cfdp_r.c.

References `CF_AppData`, `CF_CFDP_R_PDU_EOF_ERR_EID`, `CF_CFDP_R_SIZE_MISMATCH_ERR_EID`, `CF_CFDP_RecvEof()`, `CF_REC_PDU_BAD_EOF_ERROR`, `CF_REC_PDUFSIZE_MISMATCH_ERROR`, `CF_TxnState_R2`, `CFE_EVT_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_Logical_ExtHeader::eof`, `CF_HkRecv::error`, `CF_HkCounters::fault`, `CF_HkFault::file_size_mismatch`, `CF_Transaction::flags`, `CF_Transaction::fsiz`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_Logical_PduBuffer::int_header`, `CF_Flags_Rx::md_recv`, `CF_HkPacket::Payload`, `CF_HkCounters::recv`, `CF_StateFlags::rx`, `CF_History::seq_num`, `CF_Logical_PduEof::size`, `CF_History::src_eid`, and `CF_Transaction::state`.

Referenced by `CF_CFDP_R1_SubstateRecvEof()`, and `CF_CFDP_R2_SubstateRecvEof()`.

Here is the call graph for this function:



12.27.2.22 CF_CFDP_R_SubstateSendNak() `CFE_Status_t` CF_CFDP_R_SubstateSendNak (`CF_Transaction_t * txn`)

Send a NAK PDU for R2.

Description

NAK PDU is sent when there are gaps in the received data. The chunks class tracks this and generates the NAK PDU by calculating gaps internally and calling `CF_CFDP_R2_GapCompute()`. There is a special case where if a metadata PDU has not been received, then a NAK packet will be sent to request another.

Assumptions, External Events, and Notes:

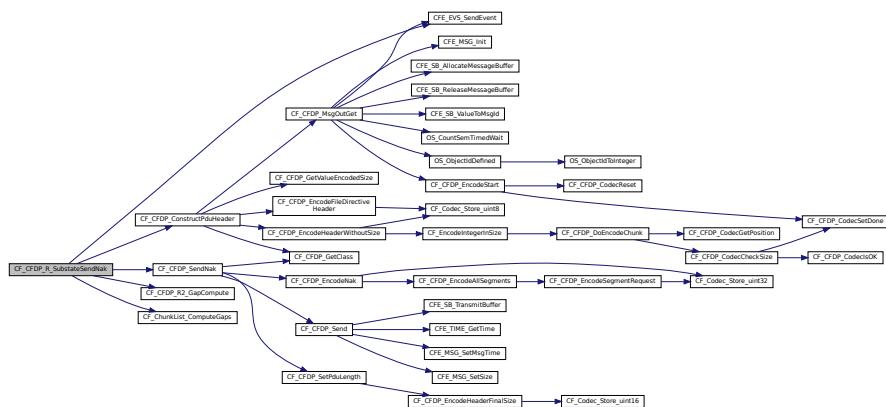
txn must not be NULL.

Return values

txn Pointer to the transaction object

References CF_AppData, CF_Assert, CF_CFDP_ConstructPduHeader(), CF_CFDP_FileDirective_NAK, CF_CFDP_R2_GapCompute(), CF_CFDP_R_REQUEST_MD_INF_EID, CF_CFDP_SendNak(), CF_ChunkList_ComputeGaps(), CF_ERROR, CF_SEND_PDU_ERROR, CF_TxnState_R2, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_ChunkList::count, CF_HkChannel_Data::counters, CF_Flags_Rx::fd_nak_sent, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_ChunkList::max_chunks, CF_Flags_Rx::md_recv, CF_Logical_IntHeader::nak, CF_HkSent::nak_segment_requests, CF_Logical_SegmentList::num_segments, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_HkPacket::Payload, CF_History::peer_eid, CF_StateFlags::rx, CF_Logical_PduNak::scope_end, CF_Logical_PduNak::scope_start, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, CF_HkCounters::sent, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Here is the call graph for this function:



```
12.27.2.23 CF_CFDP_R_Tick() void CF_CFDP_R_Tick (
```

<pre> CF_Transaction_t * txn,</pre>	<pre> int * cont)</pre>
--	-----------------------------

Perform tick (time-based) processing for R transactions.

Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send ACK, NAK, and FIN. It checks for inactivity timer and processes the ACK timer. The ACK timer is what triggers re-sends of PDUs that require acknowledgment.

Assumptions, External Events, and Notes:

txn must not be NULL. *cont* is unused, so may be NULL

Parameters

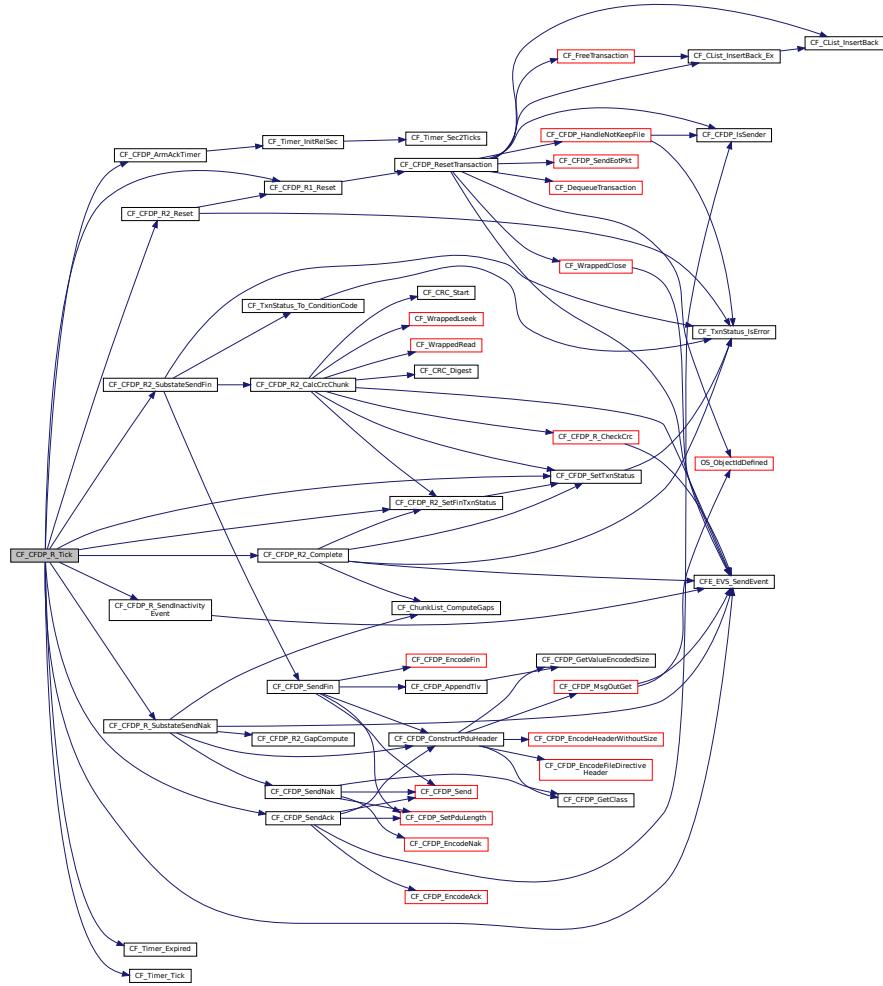
<i>txn</i>	Pointer to the transaction object
<i>cont</i>	Ignored/Unused

Definition at line 959 of file cf_cfdp_r.c.

References CF_ChannelConfig::ack_limit, CF_HkFault::ack_limit, CF_Transaction::ack_timer, CF_Flags_Common::ack_timer_armed, CF_RxS2_Data::acknak_count, CF_AppData, CF_ASSERT, CF_CFDP_AckTxnStatus_ACTIVE, CF_CFDP_ArmAckTimer(), CF_CFDP_FileDirective_EOF, CF_CFDP_R1_Reset(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SetFinTxnStatus(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_ACK_LIMIT_ERR_EID, CF_CFDP_R_SendInactivityEvent(), CF_CFDP_R_SubstateSendNak(), CF_CFDP_SendAck(), CF_CFDP_SetTxnStatus(), CF_RxSubState_WAIT_FOR_FIN_ACK, CF_SEND_PDU_ERROR, CF_SEND_PD_U_NO_BUF_AVAIL_ERROR, CF_Timer_Expired(), CF_Timer_Tick(), CF_TxnState_R2, CF_TxnStatus_ACK_LIMIT_NO_FIN, CF_TxnStatus_INACTIVITY_DETECTED, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_StateFlags::com, CF_Flags_Rx::complete, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_RxS2_Data::eof_cc, CF_HkCounters::fault, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_Flags_Rx::inactivity_fired, CF_Transaction::inactivity_timer, CF_HkPacket::Payload, CF_History::peer_eid, CF_RxState_Data::r2, CF_State_Data::receive, CF_StateFlags::rx, CF_Flags_Rx::send_ack, CF_Flags_Rx::send_fin, CF_Flags_Rx::send_nak, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, and CF_RxState_Data::sub_state.

Referenced by CF_CFDP_TickTransactions().

Here is the call graph for this function:



12.28 apps/cf/fsw/src/cf_cfdp_s.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_dispatch.h"
#include <stdio.h>
#include <string.h>
#include "cf_assert.h"
```

Functions

- static void `CF_CFDP_S_Reset (CF_Transaction_t *txn)`
- `CFE_Status_t CF_CFDP_S_SendEof (CF_Transaction_t *txn)`
Send an EOF PDU.
- void `CF_CFDP_S1_SubstateSendEof (CF_Transaction_t *txn)`
Sends an EOF for S1.
- void `CF_CFDP_S2_SubstateSendEof (CF_Transaction_t *txn)`
Triggers tick processing to send an EOF and wait for EOF-ACK for S2.
- `CFE_Status_t CF_CFDP_S_SendFileData (CF_Transaction_t *txn, uint32 foffs, uint32 bytes_to_read, uint8 calc_crc)`
Helper function to populate the PDU with file data and send it.
- void `CF_CFDP_S_SubstateSendFileData (CF_Transaction_t *txn)`
Standard state function to send the next file data PDU for active transaction.
- `CFE_Status_t CF_CFDP_S_CheckAndRespondNak (CF_Transaction_t *txn)`
Respond to a NAK by sending filedata PDUs as response.
- void `CF_CFDP_S2_SubstateSendFileData (CF_Transaction_t *txn)`
Send filedata handling for S2.
- void `CF_CFDP_S_SubstateSendMetadata (CF_Transaction_t *txn)`
Send metadata PDU.
- void `CF_CFDP_S_SubstateSendFinAck (CF_Transaction_t *txn)`
Send FIN-ACK packet for S2.
- void `CF_CFDP_S2_EarlyFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
A FIN was received before file complete, so abandon the transaction.
- void `CF_CFDP_S2_Fin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 received FIN, so set flag to send FIN-ACK.
- void `CF_CFDP_S2_Nak (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 NAK PDU received handling.
- void `CF_CFDP_S2_Nak_Arm (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 NAK handling but with arming the NAK timer.
- void `CF_CFDP_S2_WaitForEofAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 received ACK PDU in wait for EOF-ACK state.
- void `CF_CFDP_S1_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S1 receive PDU processing.
- void `CF_CFDP_S2_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 receive PDU processing.
- void `CF_CFDP_S1_Tx (CF_Transaction_t *txn)`
S1 dispatch function.
- void `CF_CFDP_S2_Tx (CF_Transaction_t *txn)`
S2 dispatch function.
- void `CF_CFDP_S_Cancel (CF_Transaction_t *txn)`
Cancel an S transaction.
- void `CF_CFDP_S_Tick (CF_Transaction_t *txn, int *cont)`
Perform tick (time-based) processing for S transactions.
- void `CF_CFDP_S_Tick_Nak (CF_Transaction_t *txn, int *cont)`
Perform NAK response for TX transactions.

12.28.1 Detailed Description

The CF Application CFDP send logic source file
Handles all CFDP engine functionality specific to TX transactions.

12.28.2 Function Documentation

12.28.2.1 CF_CFDP_S1_Recv() void CF_CFDP_S1_Recv (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

S1 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

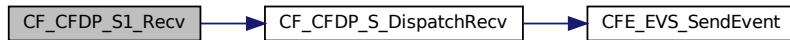
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 587 of file cf_cfdp_s.c.

References CF_CFDP_S_DispatchRecv().

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.28.2.2 CF_CFDP_S1_SubstateSendEof() void CF_CFDP_S1_SubstateSendEof (

```
CF_Transaction_t * txn )
```

Sends an EOF for S1.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

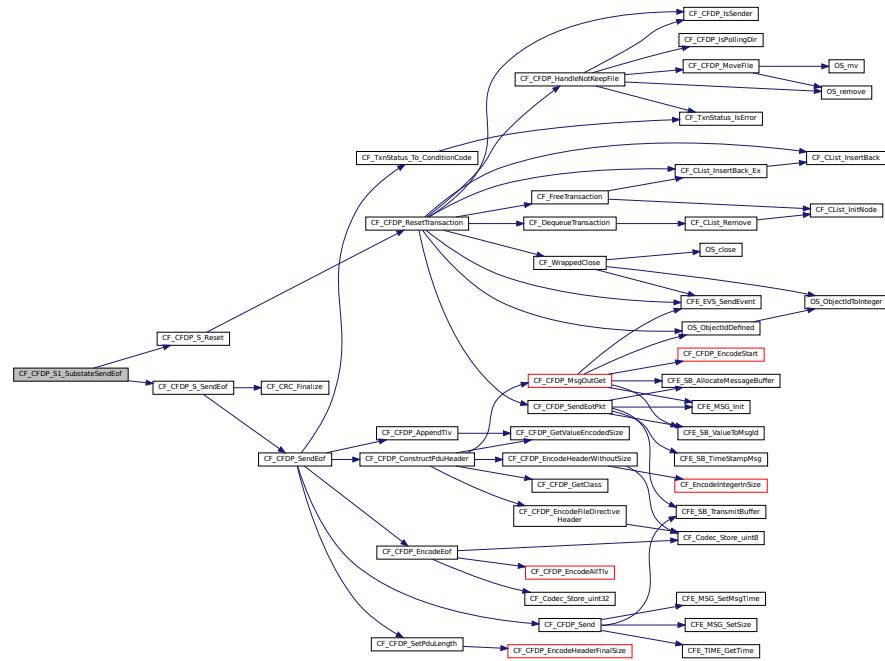
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 76 of file cf_cfdp_s.c.

References CF_CFDP_S_Reset(), CF_CFDP_S_SendEof(), and CF_SEND_PDU_NO_BUF_AVAIL_ERROR.

Referenced by CF_CFDP_S1_Tx().

Here is the call graph for this function:



12.28.2.3 CF_CFDP_S1_Tx() void CF_CFDP_S1_Tx (CF_Transaction_t * txn)

S1 dispatch function.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

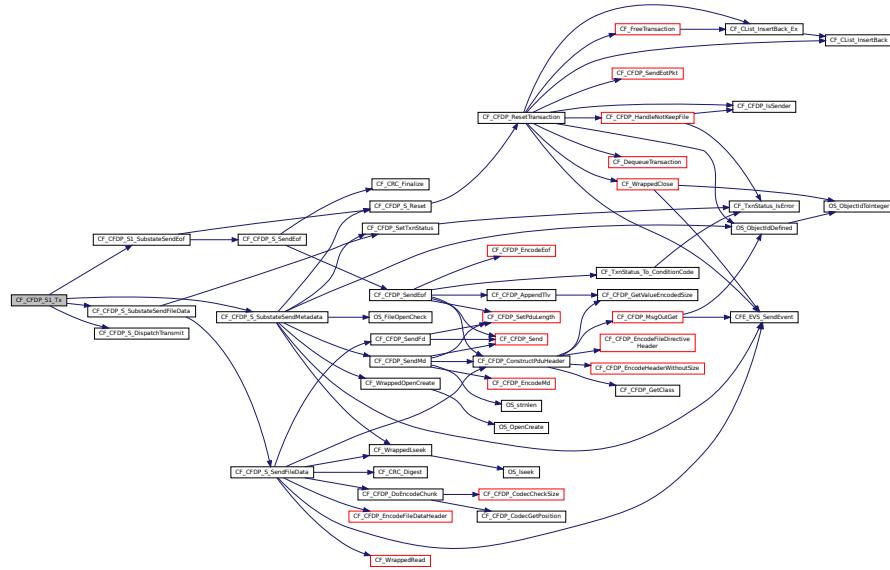
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 636 of file cf_cfdp_s.c.

References CF_CFDP_S1_SubstateSendEof(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_TxSubState_EOF, CF_TxSubState_FILEDATA, CF_TxSubState_METADATA, and CF_CFDP_S_SubstateSendDispatchTable t::substate.

Referenced by CF_CFDP_DispatchTx().

Here is the call graph for this function:



```
12.28.2.4 CF_CFDP_S2_EarlyFin() void CF_CFDP_S2_EarlyFin (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

A FIN was received before file complete, so abandon the transaction.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

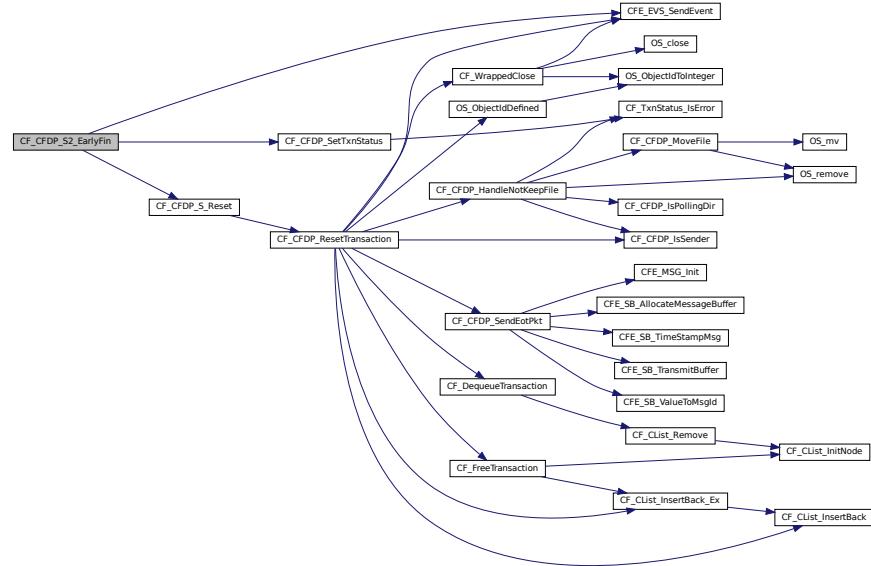
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 448 of file cf_cfdp_s.c.

References CF_CFDP_S_EARLY_FIN_ERR_EID, CF_CFDP_S_Reset(), CF_CFDP_SetTxnStatus(), CF_TxnState<_S2, CF_TxnStatus_EARLY_FIN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::history, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CE_CEDP_S2_Recy().

Here is the call graph for this function:



```
12.28.2.5 CF_CFDP_S2_Fin() void CF_CFDP_S2_Fin (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

S2 received FIN, so set flag to send FIN-ACK.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 464 of file cf_cfdp_s.c.

References CF_Logical_PduFin::cc, CF_TxSubState_SEND_FIN_ACK, CF_Logical_ExtHeader::fin, CF_TxS2_Data::fin_cc, CF_Logical_PduBuffer::int_header, CF_TxState_Data::s2, CF_StateData::send, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S2_Recv().

```
12.28.2.6 CF_CFDP_S2_Nak() void CF_CFDP_S2_Nak (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

S2 NAK PDU received handling.

Description

Stores the segment requests from the NAK packet in the chunks structure. These can be used to generate re-transmit filedata PDUs.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

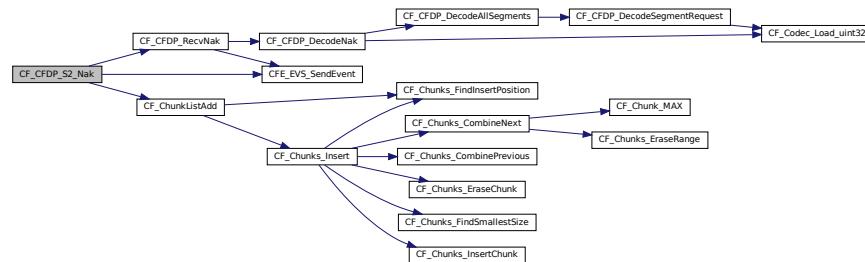
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 476 of file cf_cfdp_s.c.

References CF_AppData, CF_CFDP_RecvNak(), CF_CFDP_S_INVALID_SR_ERR_EID, CF_CFDP_S_PDU_NAK_EID, CF_ChunkListAdd(), CF_TxnState_S2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Flags_Tx::md_need_send, CF_Logical_IntHeader::nak, CF_HkRecv::nak_segment_requests, CF_Logical_SegmentList::num_segments, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_HkPacket::Payload, CF_HkCounters::recv, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_StateFlags::tx.

Referenced by CF_CFDP_S2_Nak_Arm(), and CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.28.2.7 CF_CFDP_S2_Nak_Arm() void CF_CFDP_S2_Nak_Arm (CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph)

S2 NAK handling but with arming the NAK timer.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

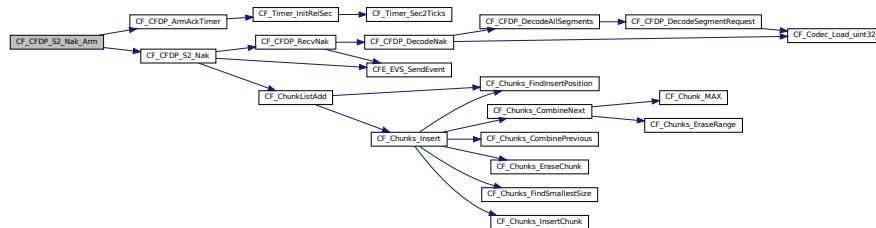
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 544 of file cf_cfdp_s.c.

References CF_CFDP_ArmAckTimer(), and CF_CFDP_S2_Nak().

Referenced by CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.28.2.8 CF_CFDP_S2_Recv()

```
void CF_CFDP_S2_Recv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

S2 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

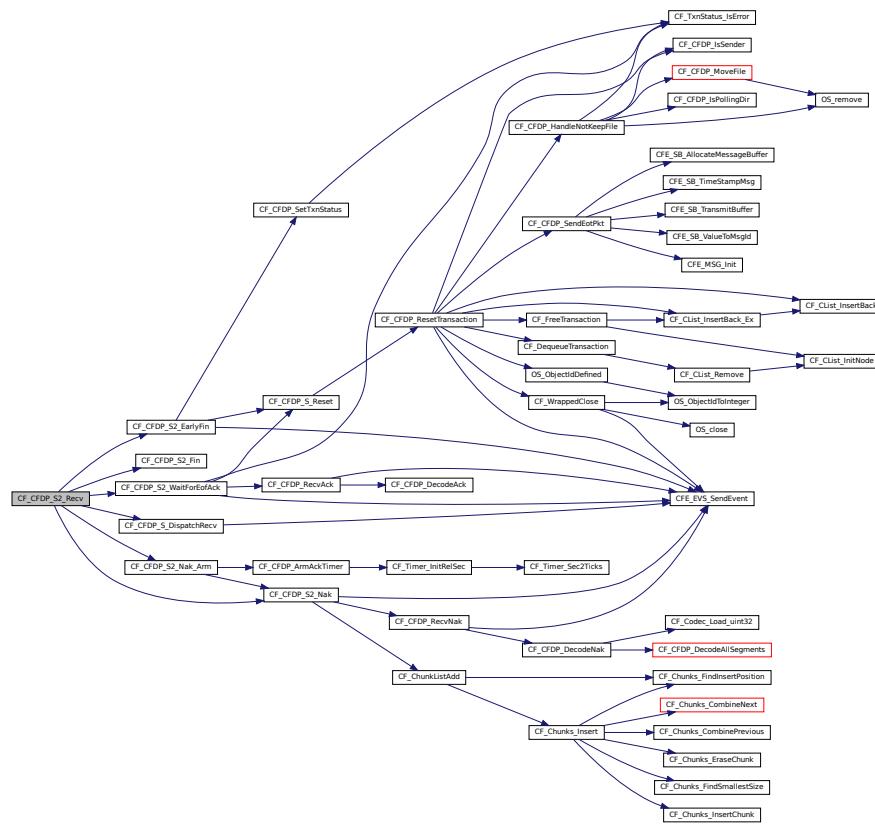
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 600 of file cf_cfdp_s.c.

References CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_FIN, CF_CFDP_FileDirective_NAK, CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Fin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_Nak_Arm(), CF_CFDP_S2_WaitForEof_Ack(), CF_CFDP_S_DispatchRecv(), CF_TxSubState_EOF, CF_TxSubState_FILEDATA, CF_TxSubState_METAD_ATA, CF_TxSubState_SEND_FIN_ACK, CF_TxSubState_WAIT_FOR_EOF_ACK, CF_TxSubState_WAIT_FOR_FIN, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, and CF_CFDP_S_SubstateRecvDispatchTable_t::substate.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.28.2.9 CF_CFDP_S2_SubstateSendEof()

```
void CF_CFDP_S2_SubstateSendEof (
    CF_Transaction_t * txn )
```

Triggers tick processing to send an EOF and wait for EOF-ACK for S2.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

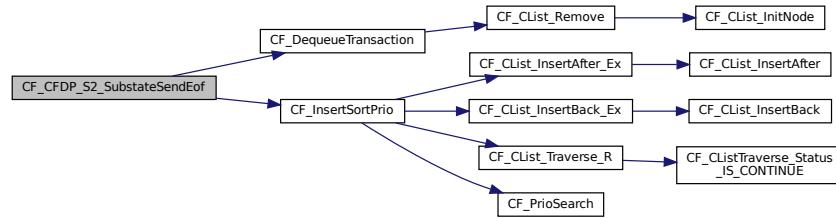
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 93 of file cf_cfdp_s.c.

References CF_Flags_Common::ack_timer_armed, CF_DequeueTransaction(), CF_InsertSortPrio(), CF_QueueIdx_← TXW, CF_TxSubState_WAIT_FOR_EOF_ACK, CF_StateFlags::com, CF_Transaction::flags, CF_StateData::send, C← F_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S2_Tx().

Here is the call graph for this function:



12.28.2.10 CF_CFDP_S2_SubstateSendFileData()

```
void CF_CFDP_S2_SubstateSendFileData (
    CF_Transaction_t * txn )
```

Send filedata handling for S2.

Description

S2 will either respond to a NAK by sending retransmits, or in absence of a NAK, it will send more of the original file data.

Assumptions, External Events, and Notes:

txn must not be NULL.

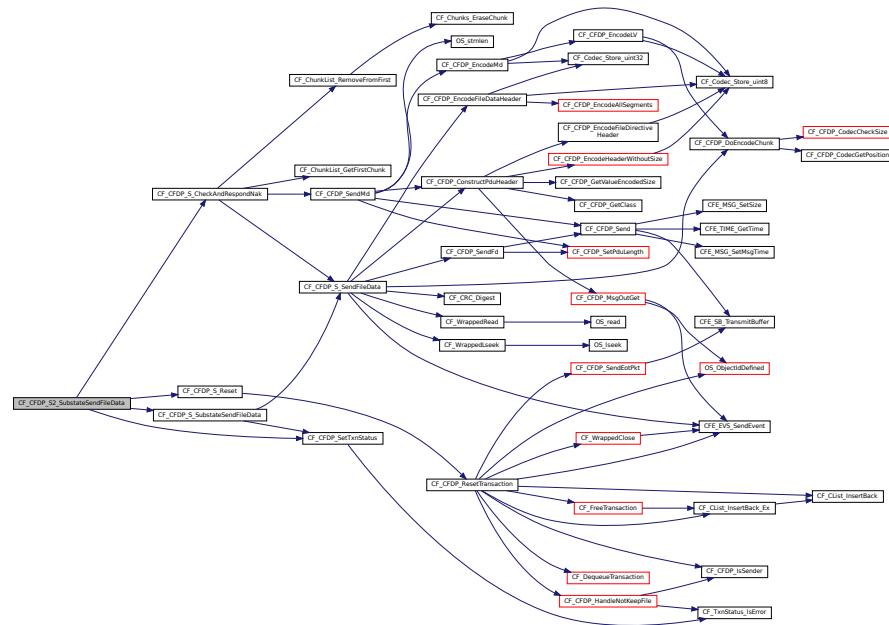
Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 304 of file cf_cfdp_s.c.

References CF_CFDP_S_CheckAndRespondNak(), CF_CFDP_S_Reset(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_SetTxnStatus(), and CF_TxnStatus_NAK_RESPONSE_ERROR.
Referenced by CF_CFDP_S2_Tx().

Here is the call graph for this function:



12.28.2.11 CF_CFDP_S2_Tx() void CF_CFDP_S2_Tx (CF_Transaction_t * *txn*)

S2 dispatch function.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

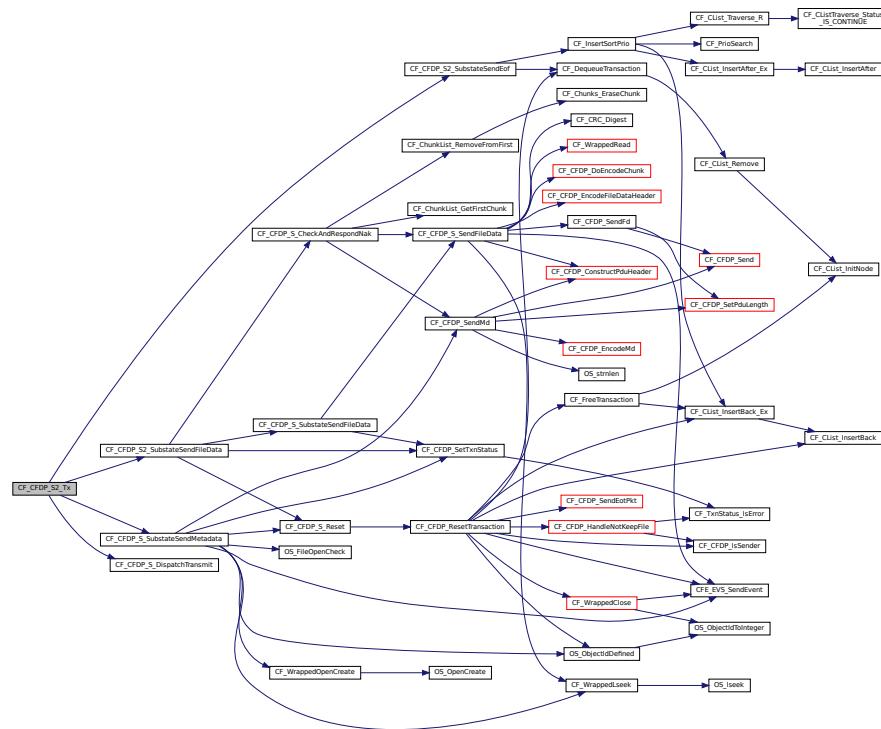
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 654 of file cf_cfdp_s.c.

References `CF_CFDP_S2_SubstateSendEof()`, `CF_CFDP_S2_SubstateSendFileData()`, `CF_CFDP_S_Dispatch`←
`Transmit()`, `CF_CFDP_S_SubstateSendMetadata()`, `CF_TxSubState_EOF`, `CF_TxSubState_FILENODATA`, `CF_TxSubState_METADATA`, and `CF_CFDP_S_SubstateSendDispatchTable_t::substate`.

Referenced by `CF_CFDP_DispatchTx()`.

Here is the call graph for this function:



12.28.2.12 CF_CFDP_S2_WaitForEofAck() void CF_CFDP_S2_WaitForEofAck (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S2 received ACK PDU in wait for EOF-ACK state.

Description

This function will trigger a state transition to CF_TxSubState_WAIT_FOR_FIN, which waits for a FIN PDU.

Assumptions, External Events, and Notes:

tx must not be NULL. ph must not be NULL.

Parameters

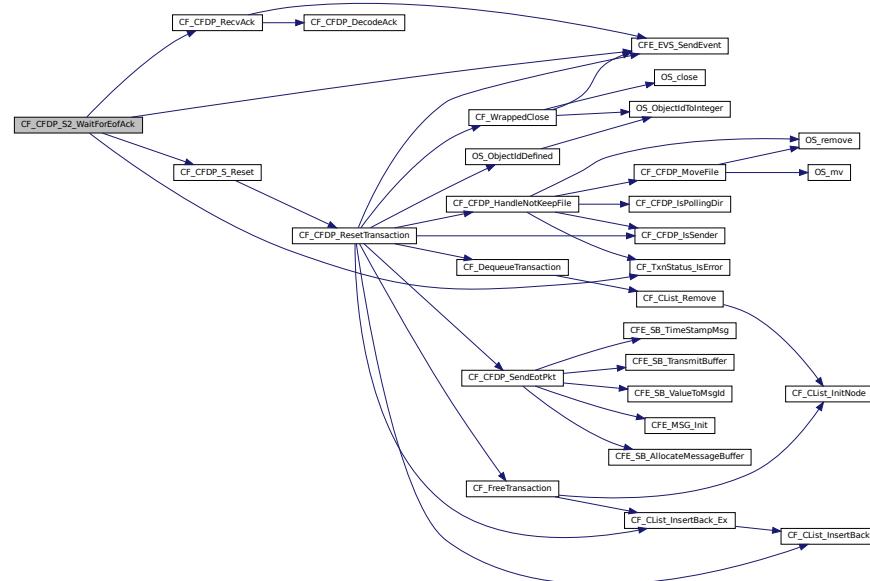
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 556 of file cf_cfdp_s.c.

References CF_Flags_Common::ack_timer_armed, CF_AppData, CF_CFDP_RecvAck(), CF_CFDP_S_PDU_EO_F_ERR_EID, CF_CFDP_S_Reset(), CF_TxnState_S2, CF_TxnStatus_IsError(), CF_TxSubState_WAIT_FOR_F_IN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_StateFlags::com, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::flags, CF

`_Transaction::history`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, `CF_HkCounters::recv`, `CF_StateData::send`, `CF_History::seq_num`, `CF_History::src_eid`, `CF_Transaction::state`, `CF_Transaction::state_data`, `CF_TxState_Data::sub_state`, and `CF_History::txn_stat`.

Referenced by CF_CFDP_S2_Recv().
Here is the call graph for this function:



12.28.2.13 CF_CFDP_S_Cancel() void CF_CFDP_S_Cancel (CF_Transaction_t * txnn)

Cancel an S transaction.

Assumptions, External Events, and Notes:

txn must not be NULL

Parameters

txn Pointer to the transaction object

Definition at line 672 of file cf_cfdp.s.c

References CF_TxSubState_EOF, CF_StateData::send, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CE_CEDP_CancelTransaction()

12.28.2.14 CF_CFDP_S_CheckAndRespondNak() CFE_Status_t CF_CFDP_S_CheckAndRespondNak (

Respond to a NAK by sending filedata PDU's as response

Description

Checks to see if a metadata PDU or filedata re-transmits must occur.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Returns

CF ERROR if error.

Return values

<i>O</i>	if no NAK processed.
<i>1</i>	if NAK processed.

Parameters

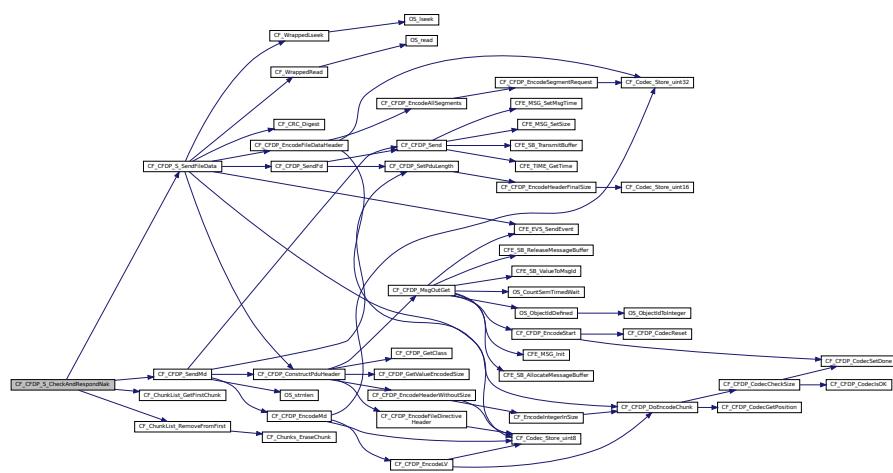
txn Pointer to the transaction object

Definition at line 249 of file cf_cfdp.s.c.

References CF_CFDP_S_SendFileData(), CF_CFDP_SendMd(), CF_ChunkList_GetFirstChunk(), CF_ChunkList_RemoveFromFirst(), CF_ERROR, CF_SEND_PDU_ERROR, CFE_SUCCESS, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Transaction::flags, CF_Flags_Tx::md_need_send, CF_Chunk::offset, CF_Chunk::size, and CF_StateFlags::tx.

Referenced by CF_CFDP_S2_SubstateSendFileData(), and CF_CFDP_S_Tick_Nak().

Here is the call graph for this function:



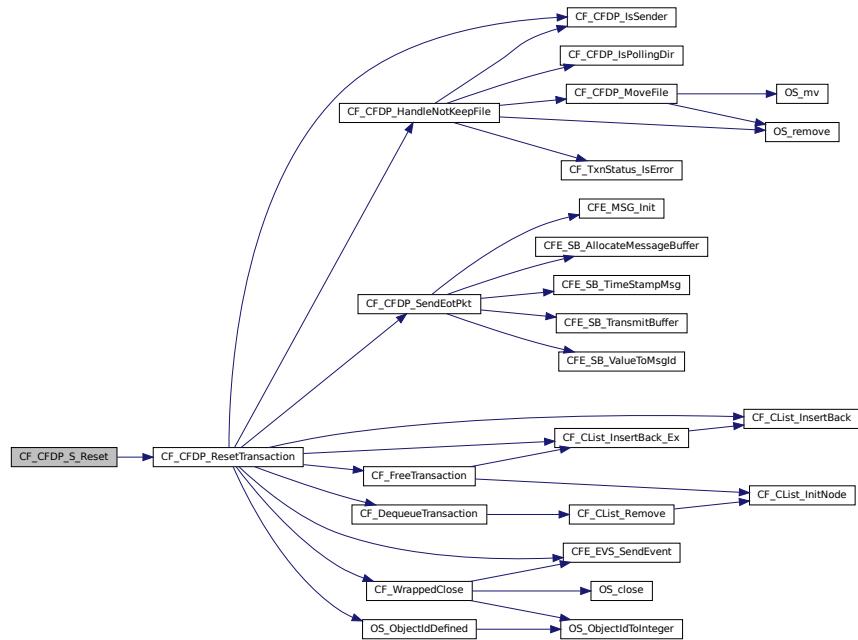
12.28.2.15 CF_CFDP_S_Reset() static void CF_CFDP_S_Reset (CF_Transaction_t * txn) [inline], [static]

Definition at line 49 of file cf_cfdp_s.c

References GE_GEDP_BesetTransaction()

Referenced by CF_CFDP_S1_SubstateSendEof(), CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_SubstateSendFileData(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_S_SubstateSendMetadata(), and CF_CFDP_S_Tick().

Here is the call graph for this function:



12.28.2.16 CF_CFDP_S_SendEof() `CFE_Status_t` `CF_CFDP_S_SendEof (`
`CF_Transaction_t * txn)`

Send an EOF PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.
<i>CF_SEND_PDU_ERROR</i>	if an error occurred while building the packet.

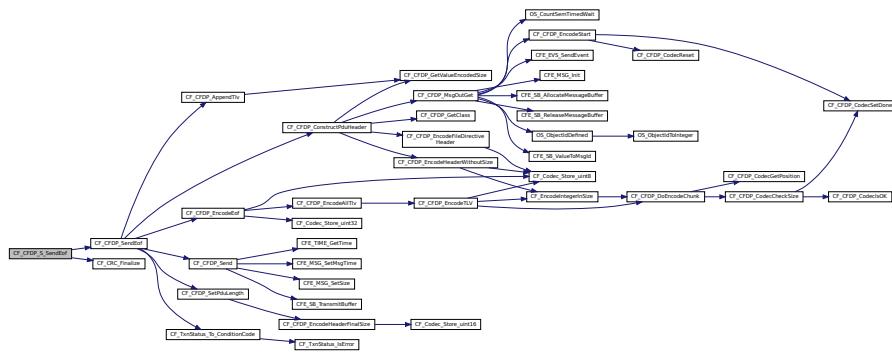
Parameters

txn Pointer to the transaction object

Definition at line 60 of file cf_cfdp_s.c.

References CF_CFDP_SendEof(), CF_CRC_Finalize(), CF_StateFlags::com, CF_Transaction::crc, CF_Flags__Common::crc_calc, and CF_Transaction::flags.

Referenced by CF_CFDP_S1_SubstateSendEof(), and CF_CFDP_S_Tick().
Here is the call graph for this function:



12.28.2.17 CF_CFDP_S_SendFileData() CFE_Status_t CF_CFDP_S_SendFileData (

```
CF_Transaction_t * txn,  
uint32 foffs,  
uint32 bytes_to_read,  
uint8 calc_crc )
```

Helper function to populate the PDU with file data and send it.

Description

This function checks the file offset cache and if the desired location is where the file offset is, it can skip a seek() call. The file is read into the filedatal PDU and then the PDU is sent.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Returns

The number of bytes sent in the file data PDU (CFE_SUCCESS, i.e. 0, if no bytes were processed), or CF_ERROR on error

Parameters

<i>txn</i>	Pointer to the transaction object
<i>foffs</i>	Position in file to send data from
<i>bytes_to_read</i>	Number of bytes to send (maximum)
<i>calc_crc</i>	Enable CRC/Checksum calculation

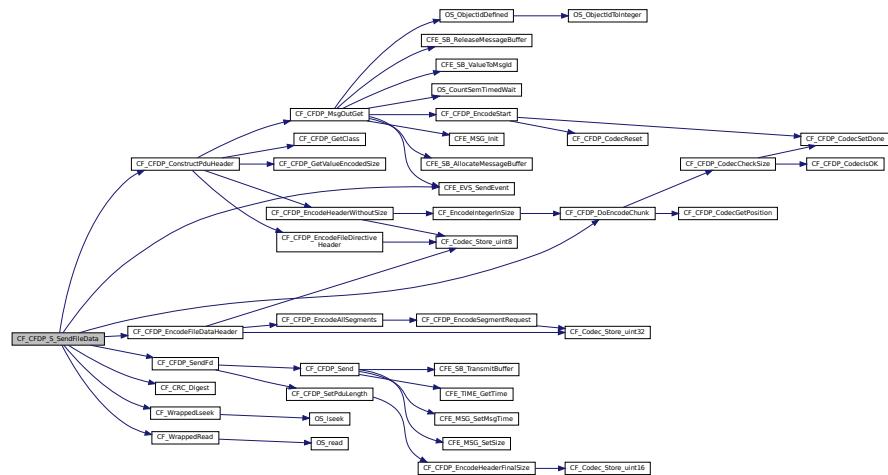
Definition at line 111 of file cf_cfdp_s.c.

References CF_TxState_Data::cached_pos, CF_AppData, CF_Assert, CF_CFDP_ConstructPduHeader(), CF_CFDP_DoEncodeChunk(), CF_CFDP_EncodeFileDataHeader(), CF_CFDP_S_READ_ERR_EID, CF_CFDP_S_SEEK_ERR_EID, CF_CFDP_SendFd(), CF_CODEC_GET_REMAIN, CF_CRC_Digest(), CF_ERROR, CF_TxnState_S2, CF_WrappedLseek(), CF_WrappedRead(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_AppData_t::config_table, CF_Hk

Channel_Data::counters, CF_Transaction::crc, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, CF_HkCounters::fault, CF_Logical_IntHeader::fd, CF_Transaction::fd, CF_HkSent::file_data_bytes, CF_HkFault::file_read, CF_HkFault::file_seek, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, C_F_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_Logical_PduFileDataHeader::offset, OS_SEEK_SET, CF_ConfigTable::outgoing_file_chunk_size, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_Logical_PduHeader::segment_meta_flag, CF_StateData::send, CF_HkCounters::sent, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_S_CheckAndRespondNak(), and CF_CFDP_S_SubstateSendFileData().

Here is the call graph for this function:



12.28.2.18 CF_CFDP_S_SubstateSendFileData()

```
void CF_CFDP_S_SubstateSendFileData (
    CF_Transaction_t * txn )
```

Standard state function to send the next file data PDU for active transaction.

Description

During the transfer of active transaction file data PDUs, the file offset is saved. This function sends the next chunk of data. If the file offset equals the file size, then transition to the EOF state.

Assumptions, External Events, and Notes:

tx must not be NULL.

Parameters

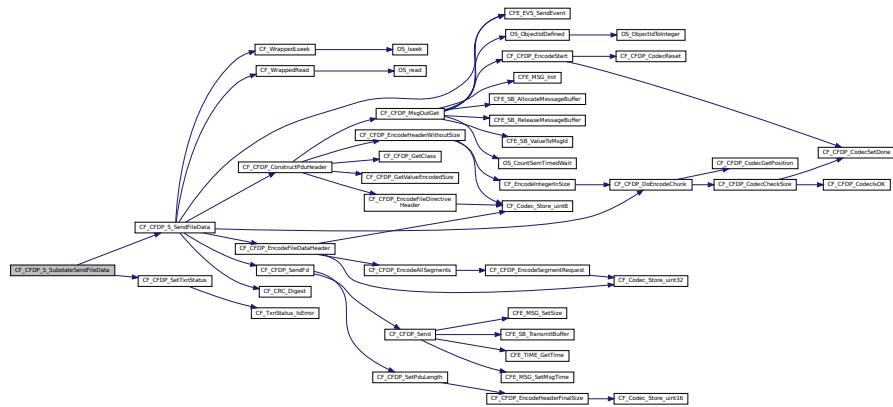
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 218 of file cf_cfdp_s.c.

References CF_CFDP_S_SendFileData(), CF_CFDP_SetTxnStatus(), CF_TxnStatus_FILESTORE_REJECTION, C_F_TxSubState_EOF, CF_Transaction::foffs, CF_Transaction::fsize, CF_StateData::send, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S1_Tx(), and CF_CFDP_S2_SubstateSendFileData().

Here is the call graph for this function:



12.28.2.19 CF_CFDP_S_SubstateSendFinAck() void CF_CFDP_S_SubstateSendFinAck (

```
CF_Transaction_t * txn )
```

Send FIN-ACK packet for S2.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

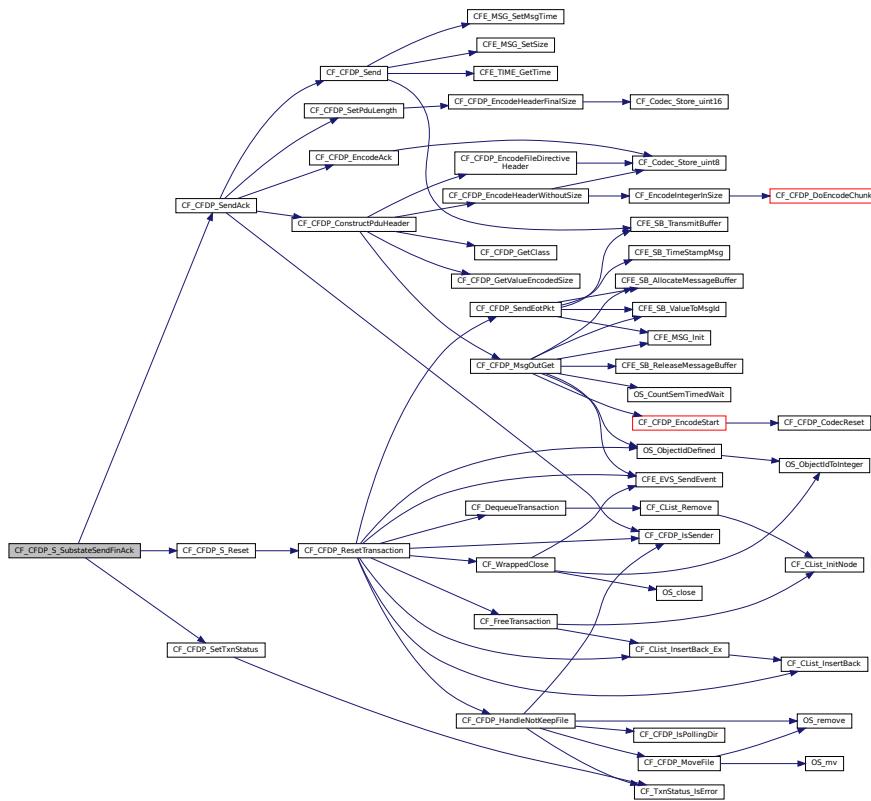
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 431 of file cf_cfdp_s.c.

References CF_CFDP_AckTxnStatus_ACTIVE, CF_CFDP_FileDirective_FIN, CF_CFDP_S_Reset(), CF_CFDP_SendAck(), CF_CFDP_SetTxnStatus(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_TxnStatus_NO_ERROR, CF_F_TxS2_Data::fin_cc, CF_Transaction::history, CF_History::peer_eid, CF_TxState_Data::s2, CF_StateData::send, CF_History::seq_num, and CF_Transaction::state_data.

Referenced by CF CFDP S Tick().

Here is the call graph for this function:



12.28.2.20 CF_CFDP_S_SubstateSendMetadata() void CF_CFDP_S_SubstateSendMetadata (CF_Transaction_t * *txn*)

Send metadata PDU.

Description

Construct and send a metadata PDU. This function determines the size of the file to put in the metadata PDU.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

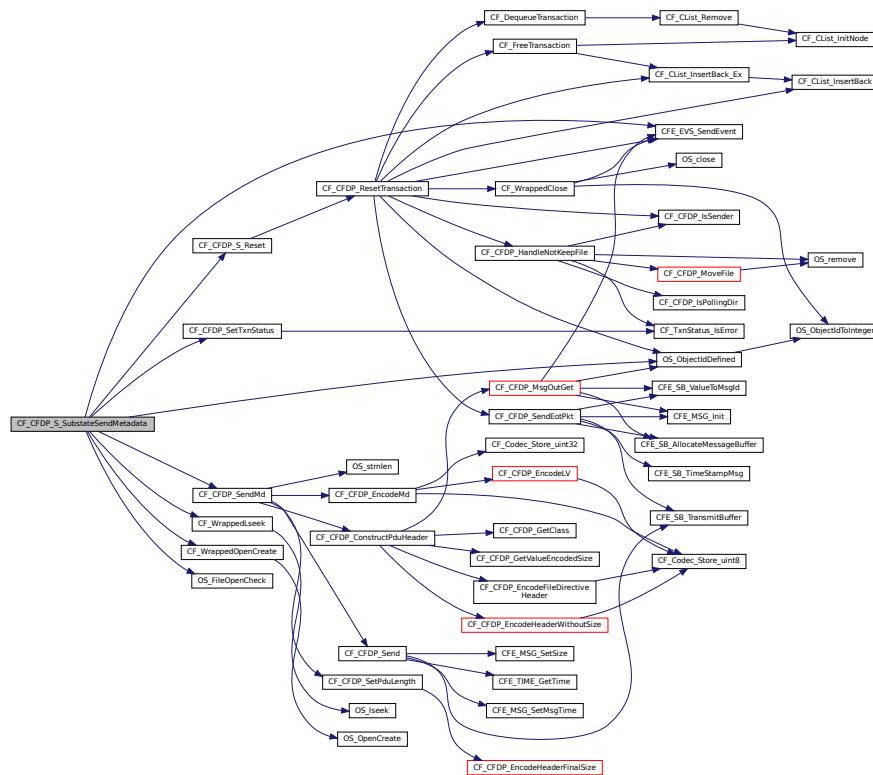
Definition at line 329 of file cf_cfdp_s.c.

References CF_AppData, CF_CFDP_S_ALREADY_OPEN_ERR_EID, CF_CFDP_S_OPEN_ERR_EID, CF_CFDP_S_Reset(), CF_CFDP_S_SEEK_BEG_ERR_EID, CF_CFDP_S_SEEK_END_ERR_EID, CF_CFDP_S_SEND_MD_ERR_EID, CF_CFDP_SendMd(), CF_CFDP_SetTxnStatus(), CF_SEND_PDU_ERROR, CF_TxnState_S2, CF_TxnStatus_FILESTORE_REJECTION, CF_TxSubState_FILEDATA, CF_WrappedLseek(), CF_WrappedOpen()

Create(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_HkFault::file_seek, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, OS_FILE_FLAG_NONE, OS_FileOpenCheck(), OS_OBJECT_ID_UNDEFINED, OS_Object::IdDefined(), OS_READ_ONLY, OS_SEEK_END, OS_SEEK_SET, OS_SUCCESS, CF_HkPacket::Payload, CF_StateData::send, CF_History::seq_num, CF_History::src_eid, CF_TxnFilenames::src_filename, CF_Transaction::state, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S1_Tx(), and CF_CFDP_S2_Tx().

Here is the call graph for this function:



12.28.2.21 CF_CFDP_S_Tick() void CF_CFDP_S_Tick (

```
CF_Transaction_t * txn,  
int * cont )
```

Perform tick (time-based) processing for S transactions.

Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send EOF or FIN-ACK. If nothing else is sent, it checks to see if a NAK retransmit must occur.

Assumptions, External Events, and Notes:

txn must not be NULL. cont is unused, so may be NULL

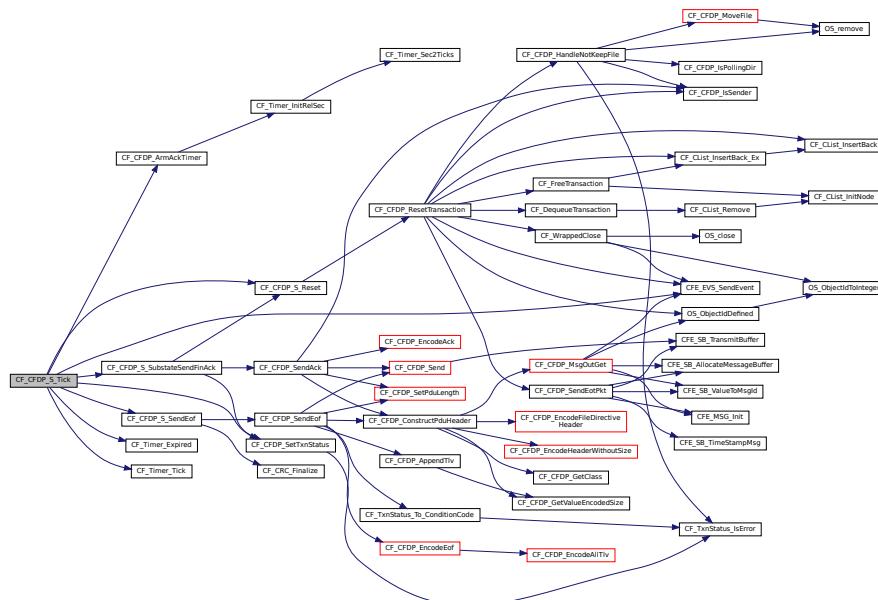
Parameters

<i>txn</i>	Pointer to the transaction object
<i>cont</i>	Unused, exists for compatibility with tick processor

Definition at line 687 of file cf_cfdp_s.c.

References CF_ChannelConfig::ack_limit, CF_HkFault::ack_limit, CF_Transaction::ack_timer, CF_Flags_Common_::ack_timer_armed, CF_TxS2_Data::acknak_count, CF_AppData, CF_CFDP_ArmAckTimer(), CF_CFDP_S_AC_K_LIMIT_ERR_EID, CF_CFDP_S_INACT_TIMER_ERR_EID, CF_CFDP_S_Reset(), CF_CFDP_S_SendEof(), CF_CFDP_S_SubstateSendFinAck(), CF_CFDP_SetTxnStatus(), CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_::Timer_Expired(), CF_Timer_Tick(), CF_TxnState_S2, CF_TxnStatus_ACK_LIMIT_NO_EOF, CF_TxnStatus_IN_ACTIVITY_DETECTED, CF_TxSubState_SEND_FIN_ACK, CF_TxSubState_WAIT_FOR_EOF_ACK, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_::Payload::channel_hk, CF_StateFlags::com, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_HkCounters::fault, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_HkFault::inactivity_timer, CF_Transaction::inactivity_timer, CF_HkPacket::Payload, CF_TxState_Data::s2, CF_StateData::send, CF_History_::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, and CF_TxState_Data::sub_state. Referenced by CF_CFDP_TickTransactions().

Here is the call graph for this function:



12.28.2.22 CF CFDP S Tick Nak() void CF CFDP S Tick Nak (

```
CF_Transaction_t * txn,  
int * cont )
```

Perform NAK response for TX transactions.

Description

This function is called at tick processing time to send pending NAK responses. It indicates "cont" is 1 if there are more responses left to send.

Assumptions, External Events, and Notes:

txn must not be NULL. cont must not be NULL.

Parameters

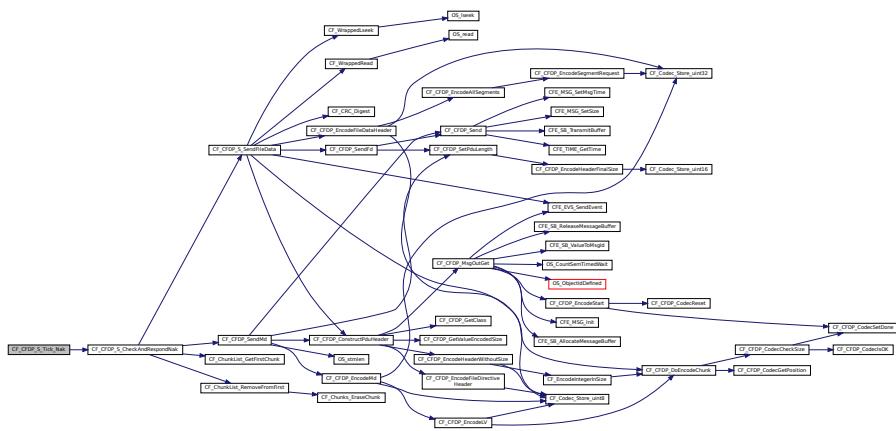
<i>txn</i>	Pointer to the transaction object
<i>cont</i>	Set to 1 if a NAK was generated

Definition at line 772 of file cf_cfdp_s.c.

References CF_CFDP_S_CheckAndRespondNak().

Referenced by CF_CFDP_TickTransactions().

Here is the call graph for this function:



12.29 apps/cf/fsw/src/cf_cfdp_s.h File Reference

```
#include "cf_cfdp_types.h"
```

Functions

- void **CF_CFDP_S1_Recv** (**CF_Transaction_t** *txn, **CF_Logical_PduBuffer_t** *ph)
S1 receive PDU processing.
 - void **CF_CFDP_S2_Recv** (**CF_Transaction_t** *txn, **CF_Logical_PduBuffer_t** *ph)
S2 receive PDU processing.
 - void **CF_CFDP_S1_Tx** (**CF_Transaction_t** *txn)
S1 dispatch function.
 - void **CF_CFDP_S2_Tx** (**CF_Transaction_t** *txn)
S2 dispatch function.
 - void **CF_CFDP_S_Tick** (**CF_Transaction_t** *txn, int *cont)
Perform tick (time-based) processing for S transactions.
 - void **CF_CFDP_S_Tick_Nak** (**CF_Transaction_t** *txn, int *cont)
Perform NAK response for TX transactions.
 - void **CF_CFDP_S_Cancel** (**CF_Transaction_t** *txn)
Cancel an S transaction.

- `CFE_Status_t CF_CFDP_S_SendEof (CF_Transaction_t *txn)`
Send an EOF PDU.
- `void CF_CFDP_S1_SubstateSendEof (CF_Transaction_t *txn)`
Sends an EOF for S1.
- `void CF_CFDP_S2_SubstateSendEof (CF_Transaction_t *txn)`
Triggers tick processing to send an EOF and wait for EOF-ACK for S2.
- `CFE_Status_t CF_CFDP_S_SendFileData (CF_Transaction_t *txn, uint32 foofs, uint32 bytes_to_read, uint8 calc_crc)`
Helper function to populate the PDU with file data and send it.
- `void CF_CFDP_S_SubstateSendFileData (CF_Transaction_t *txn)`
Standard state function to send the next file data PDU for active transaction.
- `CFE_Status_t CF_CFDP_S_CheckAndRespondNak (CF_Transaction_t *txn)`
Respond to a NAK by sending filedata PDUs as response.
- `void CF_CFDP_S2_SubstateSendFileData (CF_Transaction_t *txn)`
Send filedata handling for S2.
- `void CF_CFDP_S_SubstateSendMetadata (CF_Transaction_t *txn)`
Send metadata PDU.
- `void CF_CFDP_S_SubstateSendFinAck (CF_Transaction_t *txn)`
Send FIN-ACK packet for S2.
- `void CF_CFDP_S2_EarlyFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
A FIN was received before file complete, so abandon the transaction.
- `void CF_CFDP_S2_Fin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 received FIN, so set flag to send FIN-ACK.
- `void CF_CFDP_S2_Nak (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 NAK PDU received handling.
- `void CF_CFDP_S2_Nak_Arm (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 NAK handling but with arming the NAK timer.
- `void CF_CFDP_S2_WaitForEofAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
S2 received ACK PDU in wait for EOF-ACK state.

12.29.1 Detailed Description

Implementation related to CFDP Send File transactions

This file contains various state handling routines for transactions which are sending a file.

12.29.2 Function Documentation

12.29.2.1 `CF_CFDP_S1_Recv()` `void CF_CFDP_S1_Recv (` `CF_Transaction_t * txn,` `CF_Logical_PduBuffer_t * ph)`

S1 receive PDU processing.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 587 of file cf_cfdp_s.c.

References CF_CFDP_S_DispatchRecv().

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.29.2.2 CF_CFDP_S1_SubstateSendEof()

```
void CF_CFDP_S1_SubstateSendEof(
```

```
    CF_Transaction_t * txn )
```

Sends an EOF for S1.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

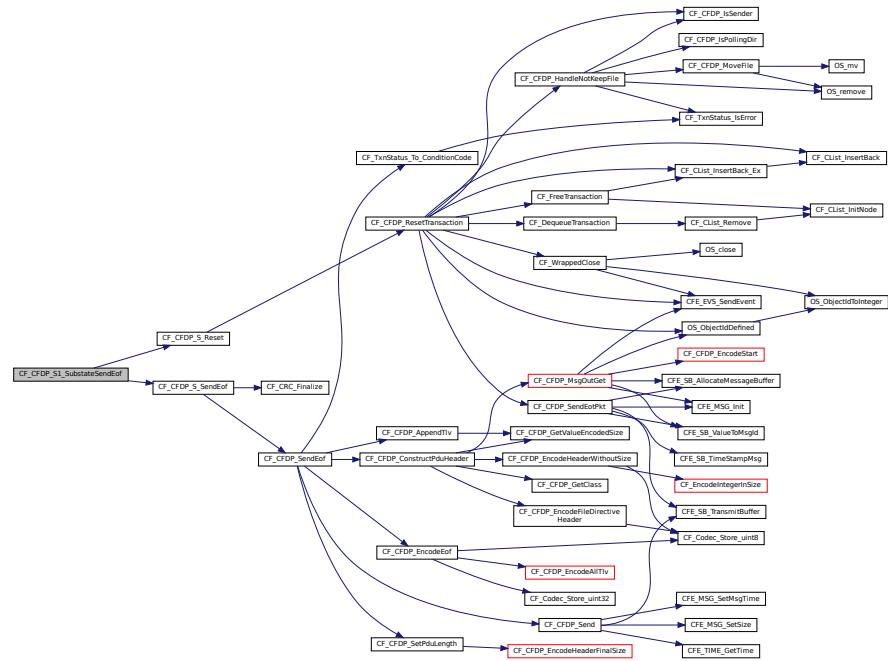
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 76 of file cf_cfdp_s.c.

References CF_CFDP_S_Reset(), CF_CFDP_S_SendEof(), and CF_SEND_PDU_NO_BUF_AVAIL_ERROR.

Referenced by CF_CFDP_S1_Tx().

Here is the call graph for this function:



12.29.2.3 CF_CFDP_S1_Tx()

```
void CF_CFDP_S1_Tx (
    CF_Transaction_t * txn )
```

S1 dispatch function.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

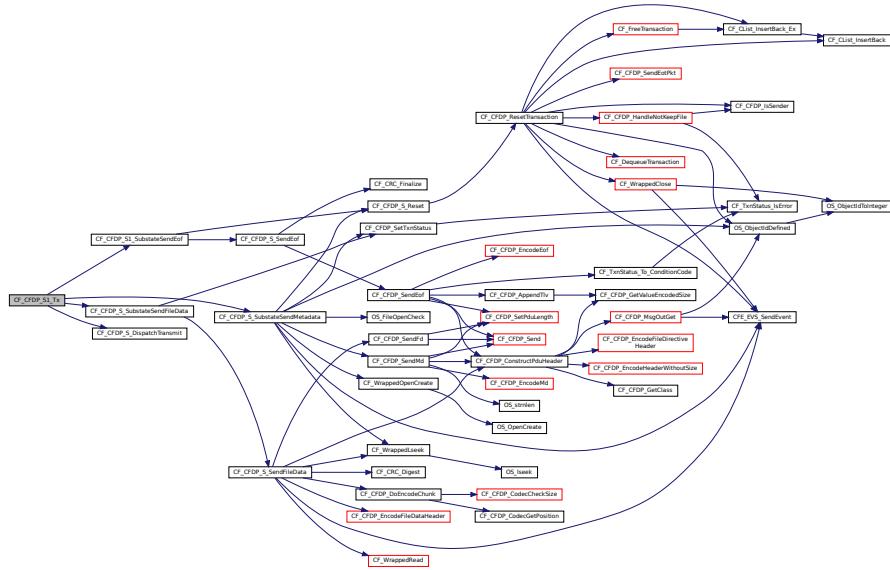
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 636 of file cf_cfdp_s.c.

References CF_CFDP_S1_SubstateSendEof(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_S_SubstateSendMetadata(), CF_TxSubState_EOF, CF_TxSubState_FILEDATA, CF_TxSubState_METADATA, and CF_CFDP_S_SubstateSendDispatchTable_t::substate.

Referenced by CF_CFDP_DispatchTx().

Here is the call graph for this function:



```
12.29.2.4 CF_CFDP_S2_EarlyFin() void CF_CFDP_S2_EarlyFin (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

A FIN was received before file complete, so abandon the transaction.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

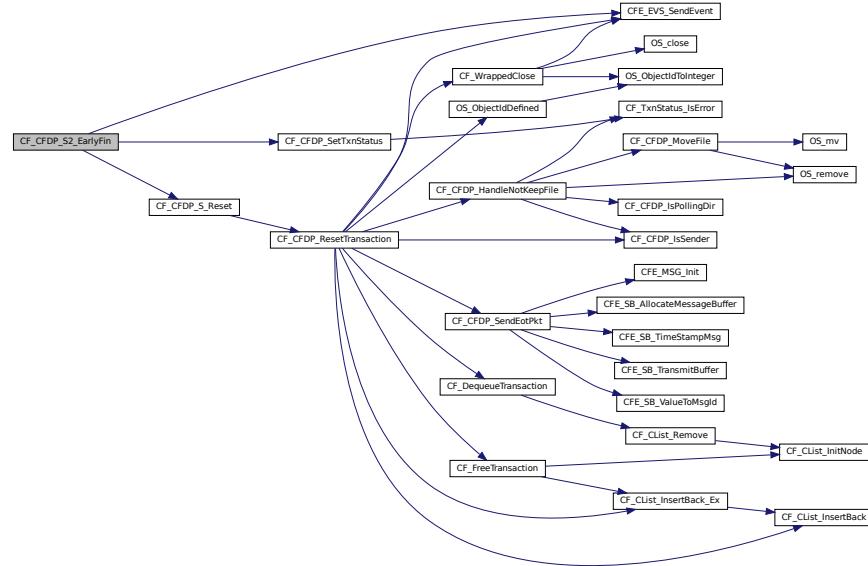
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 448 of file cf_cfdp_s.c.

References CF_CFDP_S_EARLY_FIN_ERR_EID, CF_CFDP_S_Reset(), CF_CFDP_SetTxnStatus(), CF_TxnState<_S2, CF_TxnStatus_EARLY_FIN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::history, CF_History::seq_num, CF_History::src_eid, and CF_Transaction::state.

Referenced by CE_CEDP_S2_Recy().

Here is the call graph for this function:



12.29.2.5 CF_CFDP_S2_Fin() void CF_CFDP_S2_Fin (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
  
```

S2 received FIN, so set flag to send FIN-ACK.

Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 464 of file cf_cfdp_s.c.

References CF_Logical_PduFin::cc, CF_TxSubState_SEND_FIN_ACK, CF_Logical_ImgHeader::fin, CF_TxS2_Data::fin_cc, CF_Logical_PduBuffer::int_header, CF_TxState_Data::s2, CF_StateData::send, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S2_Recv().

12.29.2.6 CF_CFDP_S2_Nak() void CF_CFDP_S2_Nak (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
  
```

S2 NAK PDU received handling.

Description

Stores the segment requests from the NAK packet in the chunks structure. These can be used to generate re-transmit filedata PDUs.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

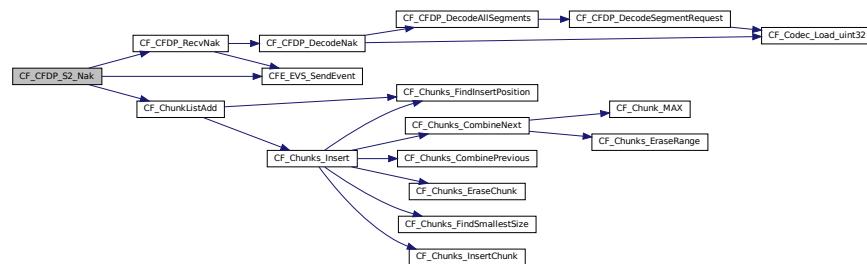
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 476 of file cf_cfdp_s.c.

References CF_AppData, CF_CFDP_RecvNak(), CF_CFDP_S_INVALID_SR_ERR_EID, CF_CFDP_S_PDU_NA-K_ERR_EID, CF_ChunkListAdd(), CF_TxnState_S2, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::flags, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_Flags_Tx::md_need_send, CF_Logical_IntHeader::nak, CF_HkRecv::nak_segment_requests, CF_Logical_SegmentList::num_segments, CF_Logical_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, CF_HkPacket::Payload, CF_HkCounters::recv, CF_Logical_PduNak::segment_list, CF_Logical_SegmentList::segments, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_StateFlags::tx.

Referenced by CF_CFDP_S2_Nak_Arm(), and CF_CFDP_S2_Recv().

Here is the call graph for this function:



```
12.29.2.7 CF_CFDP_S2_Nak_Arm() void CF_CFDP_S2_Nak_Arm (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

S2 NAK handling but with arming the NAK timer.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

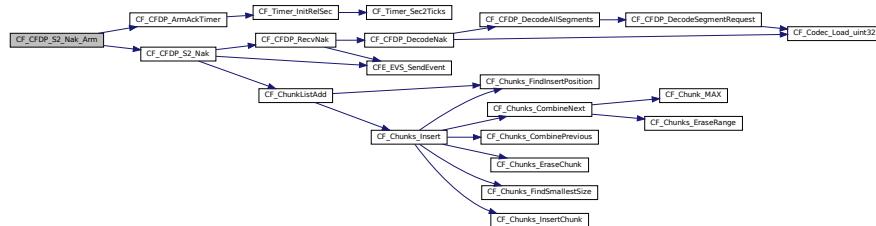
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 544 of file cf_cfdp_s.c.

References CF_CFDP_ArmAckTimer(), and CF_CFDP_S2_Nak().

Referenced by CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.29.2.8 CF_CFDP_S2_Recv()

```
void CF_CFDP_S2_Recv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

S2 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

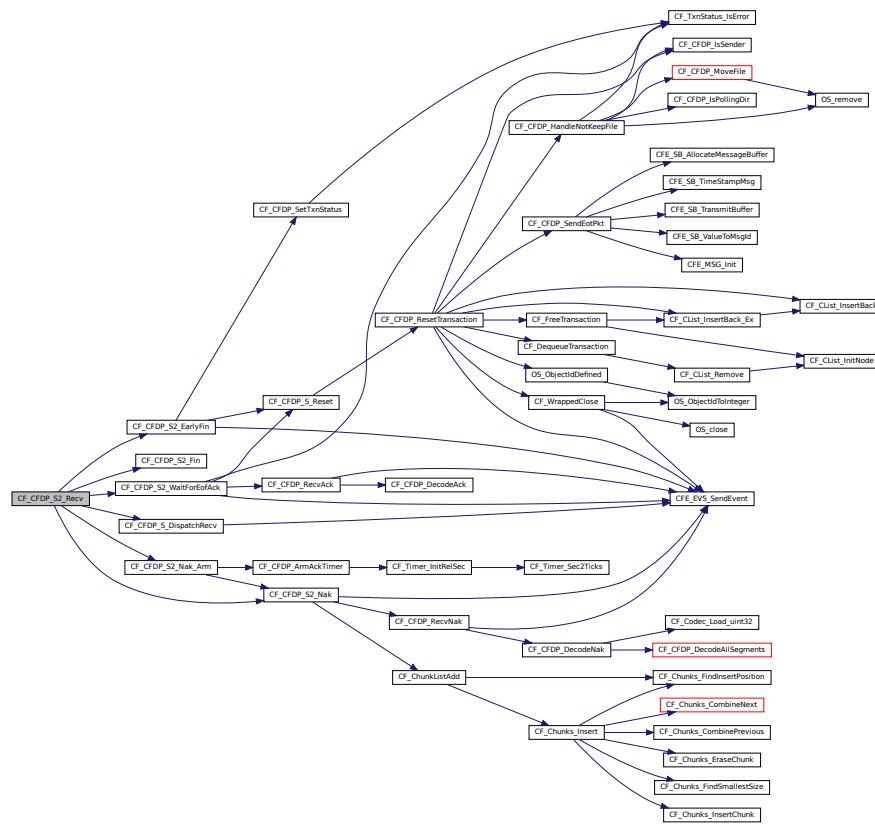
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 600 of file cf_cfdp_s.c.

References CF_CFDP_FileDirective_ACK, CF_CFDP_FileDirective_FIN, CF_CFDP_FileDirective_NAK, CF_CFDP_S2_EarlyFin(), CF_CFDP_S2_Fin(), CF_CFDP_S2_Nak(), CF_CFDP_S2_Nak_Arm(), CF_CFDP_S2_WaitForEof_Ack(), CF_CFDP_S_DispatchRecv(), CF_TxSubState_EOF, CF_TxSubState_FILEDATA, CF_TxSubState_METADATA, CF_TxSubState_SEND_FIN_ACK, CF_TxSubState_WAIT_FOR_EOF_ACK, CF_TxSubState_WAIT_FOR_FIN, CF_CFDP_FileDirectiveDispatchTable_t::fdirective, and CF_CFDP_S_SubstateRecvDispatchTable_t::substate.

Referenced by CF_CFDP_DispatchRecv().

Here is the call graph for this function:



12.29.2.9 CF_CFDP_S2_SubstateSendEof()

```
void CF_CFDP_S2_SubstateSendEof (
    CF_Transaction_t * txn )
```

Triggers tick processing to send an EOF and wait for EOF-ACK for S2.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

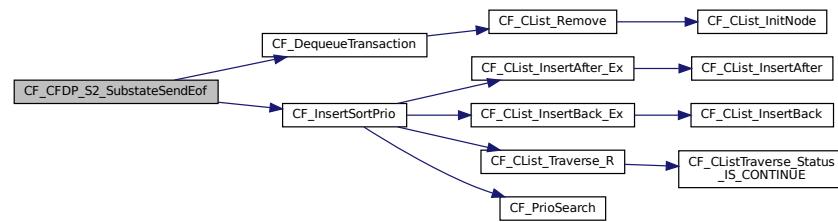
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 93 of file cf_cfdp_s.c.

References CF_Flags_Common::ack_timer_armed, CF_DequeueTransaction(), CF_InsertSortPrio(), CF_QueueIdx_← TXW, CF_TxSubState_WAIT_FOR_EOF_ACK, CF_StateFlags::com, CF_Transaction::flags, CF_StateData::send, C← F_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S2_Tx().

Here is the call graph for this function:



12.29.2.10 CF_CFDP_S2_SubstateSendFileData()

```
void CF_CFDP_S2_SubstateSendFileData (
    CF_Transaction_t * txn )
```

Send filedata handling for S2.

Description

S2 will either respond to a NAK by sending retransmits, or in absence of a NAK, it will send more of the original file data.

Assumptions, External Events, and Notes:

txn must not be NULL.

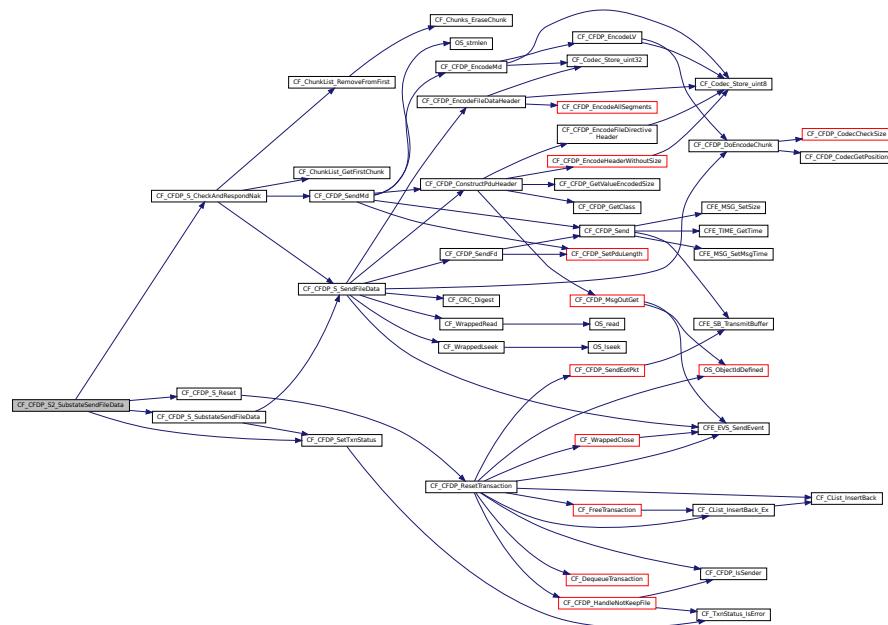
Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 304 of file cf_cfdp_s.c.

References CF_CFDP_S_CheckAndRespondNak(), CF_CFDP_S_Reset(), CF_CFDP_S_SubstateSendFileData(), CF_CFDP_SetTxnStatus(), and CF_TxnStatus_NAK_RESPONSE_ERROR.
Referenced by CF_CFDP_S2_Tx().

Here is the call graph for this function:



12.29.2.11 CF_CFDP_S2_Tx() void CF_CFDP_S2_Tx (CF_Transaction_t * *txn*)

S2 dispatch function.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

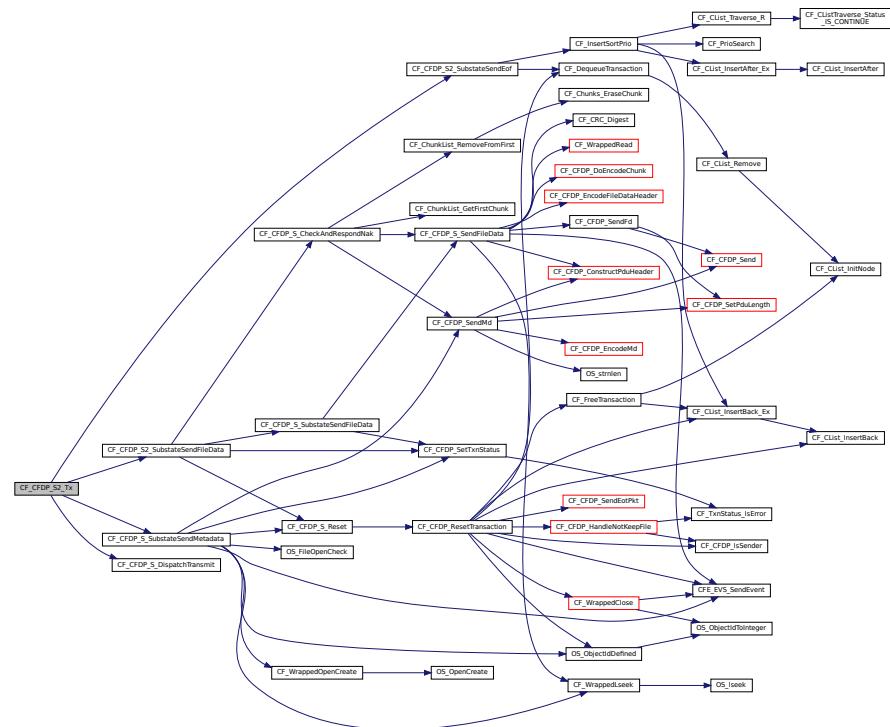
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 654 of file cf_cfdp_s.c.

References CF_CFDP_S2_SubstateSendEof(), CF_CFDP_S2_SubstateSendFileData(), CF_CFDP_S_DispatchTransmit(), CF_CFDP_S_SubstateSendMetadata(), CF_TxSubState_EOF, CF_TxSubState_FILEDATA, CF_TxSubState_METADATA, and CF_CFDP_S_SubstateSendDispatchTable_t::substate.

Referenced by CF_CFDP_DispatchTx().

Here is the call graph for this function:



12.29.2.12 CF_CFDP_S2_WaitForEofAck() void CF_CFDP_S2_WaitForEofAck (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S2 received ACK PDU in wait for EOF-ACK state.

Description

This function will trigger a state transition to CF_TxSubState_WAIT_FOR_FIN, which waits for a FIN PDU.

Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

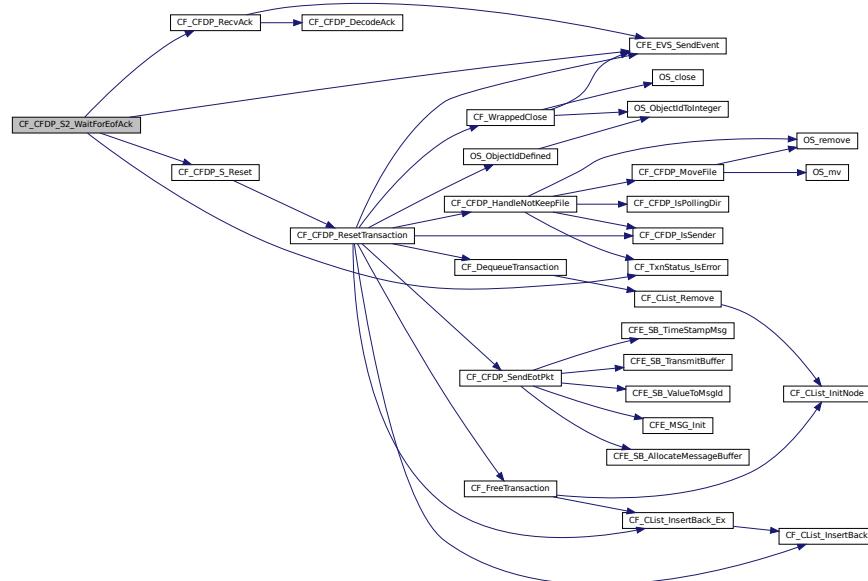
Definition at line 556 of file cf_cfdp_s.c.

References CF_Flags_Common::ack_timer_armed, CF_AppData, CF_CFDP_RecvAck(), CF_CFDP_S_PDU_EO_F_ERR_EID, CF_CFDP_S_Reset(), CF_TxnState_S2, CF_TxnStatus_IsError(), CF_TxSubState_WAIT_FOR_F_IN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Transaction::chan_num, CF_HkPacket_Payload::channel hk, CF_StateFlags::com, CF_HkChannel_Data::counters, CF_HkRecv::error, CF_Transaction::flags, CF

_Transaction::history, CF_AppData_t::hk, CF_HkPacket::Payload, CF_HkCounters::recv, CF_StateData::send, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, CF_Transaction::state_data, CF_TxState_Data::sub_state, and CF_History::txn_stat.

Referenced by CF_CFDP_S2_Recv().

Here is the call graph for this function:



12.29.2.13 CF_CFDP_S_Cancel() void CF_CFDP_S_Cancel (CF_Transaction_t * *txn*)

Cancel an S transaction.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 672 of file cf_cfdp_s.c.

References CF_TxSubState_EOF, CF_StateData::send, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_CancelTransaction().

12.29.2.14 CF_CFDP_S_CheckAndRespondNak() CFE_Status_t CF_CFDP_S_CheckAndRespondNak (CF_Transaction_t * *txn*)

Respond to a NAK by sending filedata PDUs as response.

Description

Checks to see if a metadata PDU or filedata re-transmits must occur.

Assumptions, External Events, and Notes:

txn must not be NULL.

Returns

CF_ERROR if error.

Return values

<i>O</i>	if no NAK processed.
<i>1</i>	if NAK processed.

Parameters

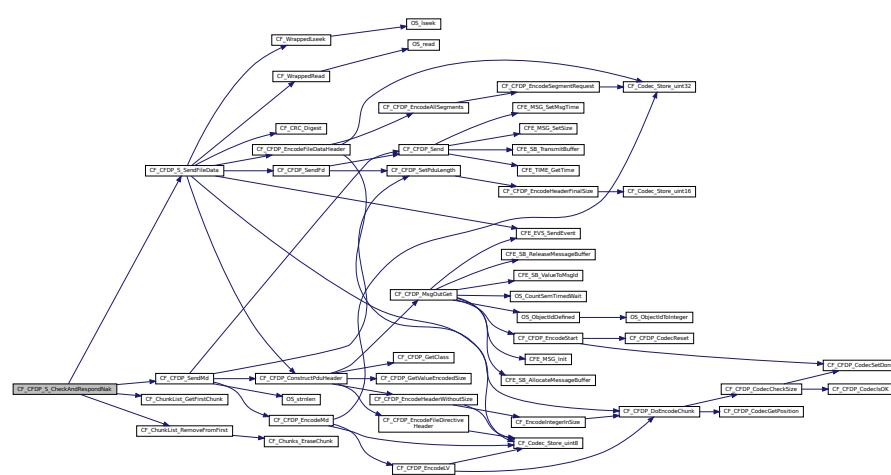
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 249 of file cf_cfdp.s.c.

References CF_CFDP_S_SendFileData(), CF_CFDP_SendMd(), CF_ChunkList_GetFirstChunk(), CF_ChunkList_RemoveFromFirst(), CF_ERROR, CF_SEND_PDU_ERROR, CFE_SUCCESS, CF_ChunkWrapper::chunks, CF_Transaction::chunks, CF_Transaction::flags, CF_Flags_Tx::md_need_send, CF_Chunk::offset, CF_Chunk::size, and CF_StateFlags::tx.

Referenced by CF_CFDP_S2_SubstateSendFileData(), and CF_CFDP_S_Tick_Nak().

Here is the call graph for this function:



12.29.2.15 CF_CFDP_S_SendEof() `CFE_Status_t CF_CFDP_S_SendEof (`
`CF_Transaction_t * txp)`

Send an EOF PDU

Assumptions, External Events, and Notes:

txn must not be NULL.

Return values

<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.
<i>CF_SEND_PDU_ERROR</i>	if an error occurred while building the packet.

Parameters

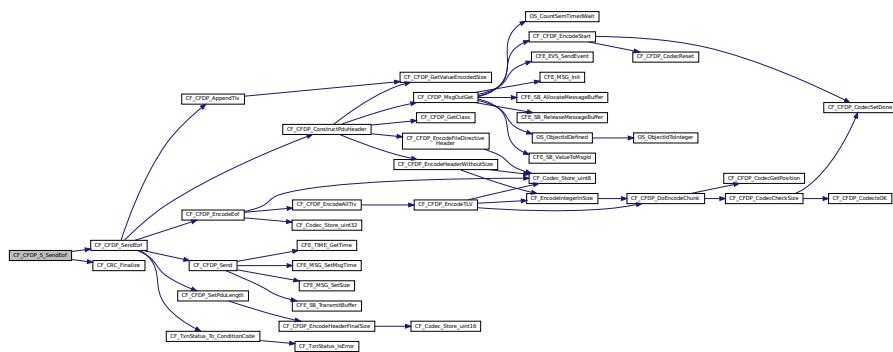
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 60 of file cf_cfdp_s.c.

References CF_CFDP_SendEof(), CF_CRC_Finalize(), CF_StateFlags::com, CF_Transaction::crc, CF_Flags_common::crc_calc, and CF_Transaction::flags.

Referenced by CF_CFDP_S1_SubstateSendEof(), and CF_CFDP_S_Tick().

Here is the call graph for this function:



12.29.2.16 CF_CFDP_S_SendFileData() CFE_Status_t CF_CFDP_S_SendFileData (

```
CF_Transaction_t * txn,  
uint32 foffs,  
uint32 bytes_to_read,  
uint8 calc_crc )
```

Helper function to populate the PDU with file data and send it.

Description

This function checks the file offset cache and if the desired location is where the file offset is, it can skip a seek() call. The file is read into the filedatal PDU and then the PDU is sent.

Assumptions, External Events, and Notes:

txn must not be NULL.

Returns

The number of bytes sent in the file data PDU (CFE_SUCCESS, i.e. 0, if no bytes were processed), or CF_ERROR on error

Parameters

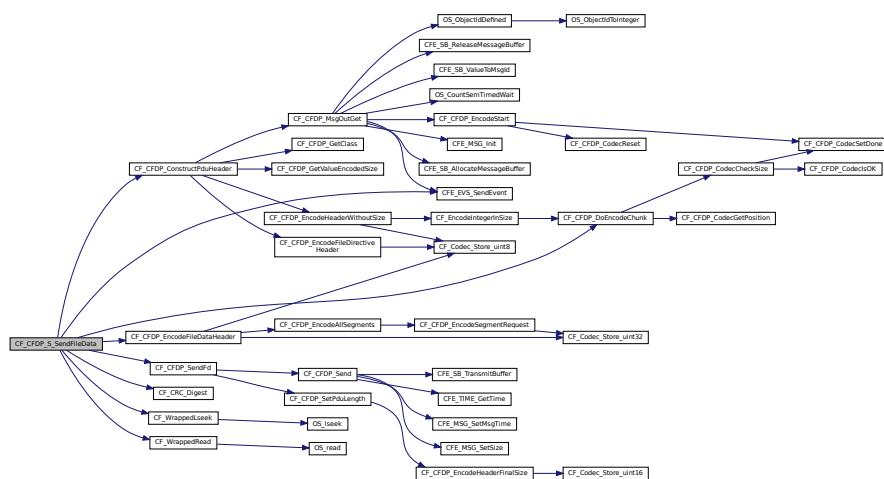
<i>txn</i>	Pointer to the transaction object
<i>foffs</i>	Position in file to send data from
<i>bytes_to_read</i>	Number of bytes to send (maximum)
<i>calc_crc</i>	Enable CRC/Checksum calculation

Definition at line 111 of file cf_cfdp_s.c.

References CF_TxState_Data::cached_pos, CF_AppData, CF_ASSERT, CF_CFDP_ConstructPduHeader(), CF_CFDP_DoEncodeChunk(), CF_CFDP_EncodeFileDataHeader(), CF_CFDP_S_READ_ERR_EID, CF_CFDP_S_SEEK, _FD_ERR_EID, CF_CFDP_SendFd(), CF_CODEC_GET_REMAIN, CF_CRC_Digest(), CF_ERROR, CF_TxnState, _S2, CF_WrappedLseek(), CF_WrappedRead(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_Transaction::crc, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, CF_HkCounters::fault, CF_Logical_IntHeader::fd, CF_Transaction::fd, CF_HkSent::file_data_bytes, CF_HkFault::file_read, CF_HkFault::file_seek, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, CF_FLogical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_Logical_PduFileDataHeader::offset, OS_SEEK_SET, CF_ConfigTable::outgoing_file_chunk_size, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_Logical_PduHeader::segment_meta_flag, CF_StateData::send, CF_HkCounters::sent, CF_History::seq_num, CF_History::src_eid, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_S_CheckAndRespondNak(), and CF_CFDP_S_SubstateSendFileData().

Here is the call graph for this function:



12.29.2.17 CF_CFDP_S_SubstateSendFileData() void CF_CFDP_S_SubstateSendFileData (CF_Transaction t * txn)

Standard state function to send the next file data PDU for active transaction.

Description

During the transfer of active transaction file data PDUs, the file offset is saved. This function sends the next chunk of data. If the file offset equals the file size, then transition to the EOF state.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

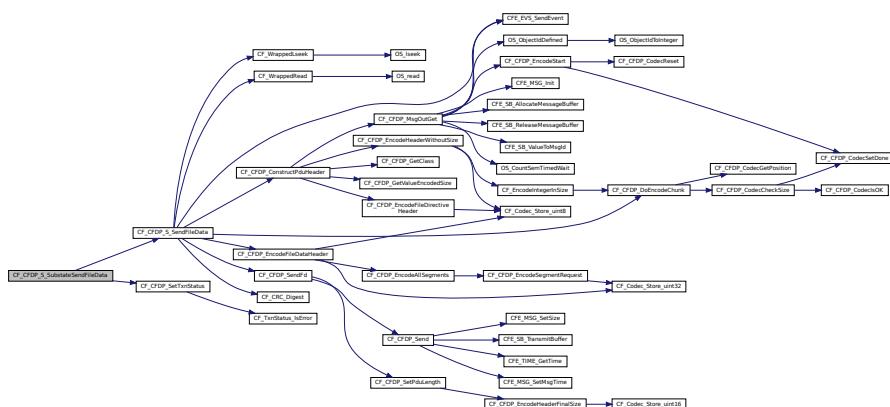
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 218 of file cf_cfdp_s.c.

References `CF_CFDP_S_SendFileData()`, `CF_CFDP_SetTxnStatus()`, `CF_TxnStatus_FILESTORE_REJECTION`, `C_F_TxSubState_EOF`, `CF_Transaction::foffs`, `CF_Transaction::fsize`, `CF_StateData::send`, `CF_Transaction::state_data`, and `CF_TxState_Data::sub_state`.

Referenced by `CF_CFDP_S1_Tx()`, and `CF_CFDP_S2_SubstateSendFileData()`.

Here is the call graph for this function:



12.29.2.18 CF_CFDP_S_SubstateSendFinAck()

```
void CF_CFDP_S_SubstateSendFinAck (
    CF_Transaction_t * txn )
```

Send FIN-ACK packet for S2.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

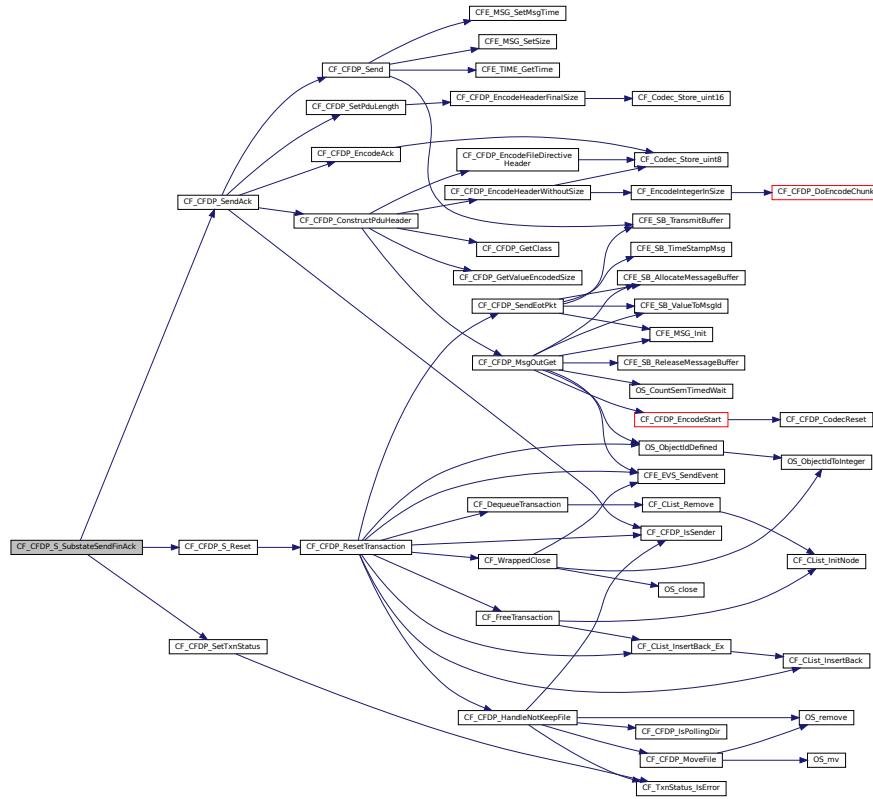
Definition at line 431 of file cf_cfdp_s.c.

References `CF_CFDP_AckTxnStatus_ACTIVE`, `CF_CFDP_FileDirective_FIN`, `CF_CFDP_S_Reset()`, `CF_CFDP_SendAck()`, `CF_CFDP_SetTxnStatus()`, `CF_SEND_PDU_NO_BUF_AVAIL_ERROR`, `CF_TxnStatus_NO_ERROR`, `C_F_TxSubState_EOF`, `CF_TxnStatus_SET_ERROR`, and `CF_TxnStatus_SET_NO_ERROR`.

F_TxS2_Data::fin_cc, CF_Transaction::history, CF_History::peer_eid, CF_TxState_Data::s2, CF_StateData::send, C_F_History::seq_num, and CF_Transaction::state_data.

Referenced by CF_CFDP_S_Tick().

Here is the call graph for this function:



12.29.2.19 CF_CFDP_S_SubstateSendMetadata() void CF_CFDP_S_SubstateSendMetadata (CF_Transaction_t * txn)

Send metadata PDU.

Description

Construct and send a metadata PDU. This function determines the size of the file to put in the metadata PDU.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

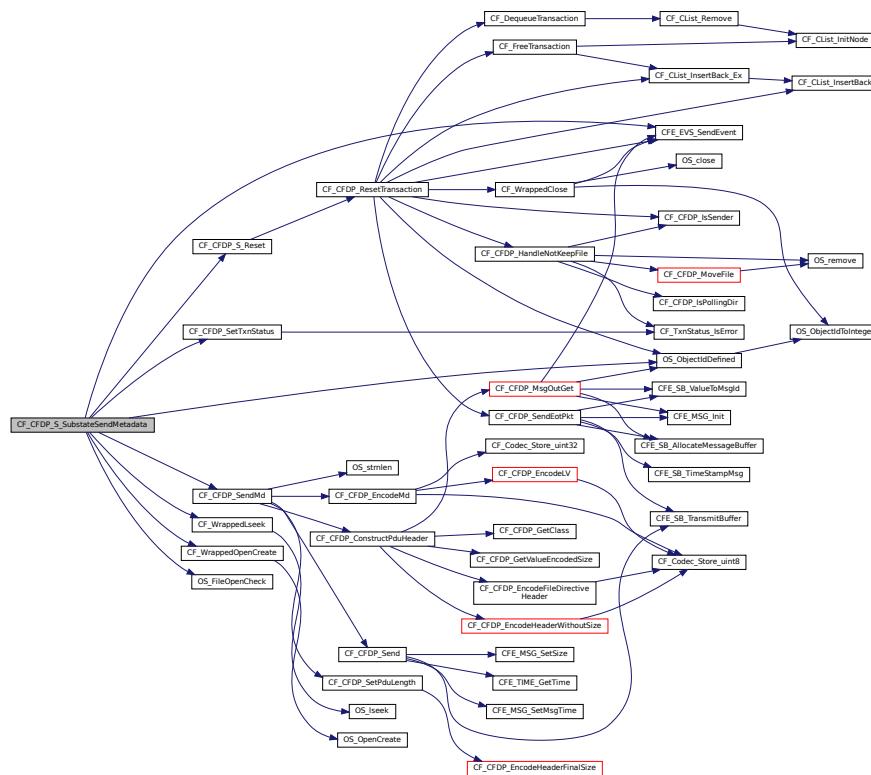
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 329 of file cf_cfdp_s.c.

DP_S_Reset(), CF_CFDP_S_SEEK_BEG_ERR_EID, CF_CFDP_S_SEEK_END_ERR_EID, CF_CFDP_S_SEN_D_MD_ERR_EID, CF_CFDP_SendMd(), CF_CFDP_SetTxnStatus(), CF_SEND_PDU_ERROR, CF_TxnState_S2, CF_TxnStatus_FILESTORE_REJECTION, CF_TxSubState_FILEDATA, CF_WrappedLseek(), CF_WrappedOpenCreate(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_HkCounters::fault, CF_Transaction::fd, CF_HkFault::file_open, CF_HkFault::file_seek, CF_History::fnames, CF_Transaction::fsize, CF_Transaction::history, CF_AppData_t::hk, OS_FILE_FLAG_NONE, OS_FileOpenCheck(), OS_OBJECT_ID_UNDEFINED, OS_ObjectIdDefined(), OS_READ_ONLY, OS_SEEK_END, OS_SEEK_SET, OS_SUCCESS, CF_HkPacket::Payload, CF_StateData::send, CF_History::seq_num, CF_History::src_eid, CF_TxnFilenames::src_filename, CF_Transaction::state, CF_Transaction::state_data, and CF_TxState_Data::sub_state.

Referenced by CF_CFDP_S1_Tx(), and CF_CFDP_S2_Tx().

Here is the call graph for this function:



```
12.29.2.20 CF_CFDP_S_Tick() void CF_CFDP_S_Tick (
    CFE_Transaction_t * txn,
    int * cont )
```

Perform tick (time-based) processing for S transactions.

Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send EOF or FIN-ACK. If nothing else is sent, it checks to see if a NAK retransmit must occur.

Assumptions, External Events, and Notes:

`txn` must not be NULL. `cont` is unused, so may be NULL

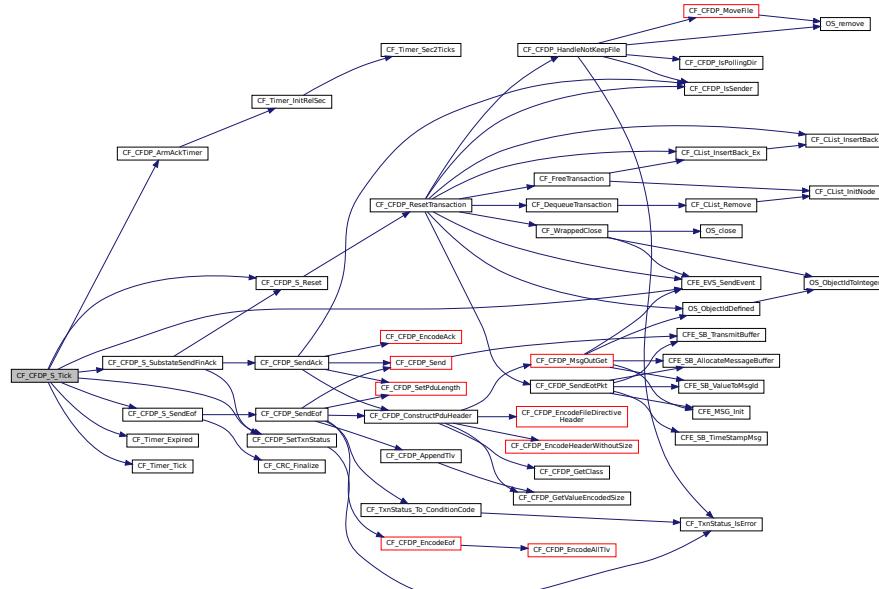
Parameters

<code>txn</code>	Pointer to the transaction object
<code>cont</code>	Unused, exists for compatibility with tick processor

Definition at line 687 of file cf_cfdp_s.c.

References `CF_ChannelConfig::ack_limit`, `CF_HkFault::ack_limit`, `CF_Transaction::ack_timer`, `CF_Flags_Common::ack_timer_armed`, `CF_TxS2_Data::acknak_count`, `CF_AppData`, `CF_CFDP_ArmAckTimer()`, `CF_CFDP_S_AC_K_LIMIT_ERR_EID`, `CF_CFDP_S_INACT_TIMER_ERR_EID`, `CF_CFDP_S_Reset()`, `CF_CFDP_S_SendEof()`, `C_F_CFDP_S_SubstateSendFinAck()`, `CF_CFDP_SetTxnStatus()`, `CF_SEND_PDU_NO_BUF_AVAIL_ERROR`, `CF_Timer_Expired()`, `CF_Timer_Tick()`, `CF_TxnState_S2`, `CF_TxnStatus_ACK_LIMIT_NO_EOF`, `CF_TxnStatus_IN_ACTIVITY_DETECTED`, `CF_TxSubState_SEND_FIN_ACK`, `CF_TxSubState_WAIT_FOR_EOF_ACK`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CF_ConfigTable::chan`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_StateFlags::com`, `CF_AppData_t::config_table`, `CF_HkChannel_Data::counters`, `CF_HkCounters::fault`, `CF_Transaction::flags`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_HkFault::inactivity_timer`, `CF_Transaction::inactivity_timer`, `CF_HkPacket::Payload`, `CF_TxState_Data::s2`, `CF_StateData::send`, `CF_History::seq_num`, `CF_History::src_eid`, `CF_Transaction::state`, `CF_Transaction::state_data`, and `CF_TxState_Data::sub_state`. Referenced by `CF_CFDP_TickTransactions()`.

Here is the call graph for this function:



```
12.29.2.21 CF_CFDP_S_Tick_Nak() void CF_CFDP_S_Tick_Nak (
    CF_Transaction_t * txn,
    int * cont )
```

Perform NAK response for TX transactions.

Description

This function is called at tick processing time to send pending NAK responses. It indicates "cont" is 1 if there are more responses left to send.

Assumptions, External Events, and Notes:

txn must not be NULL. cont must not be NULL.

Parameters

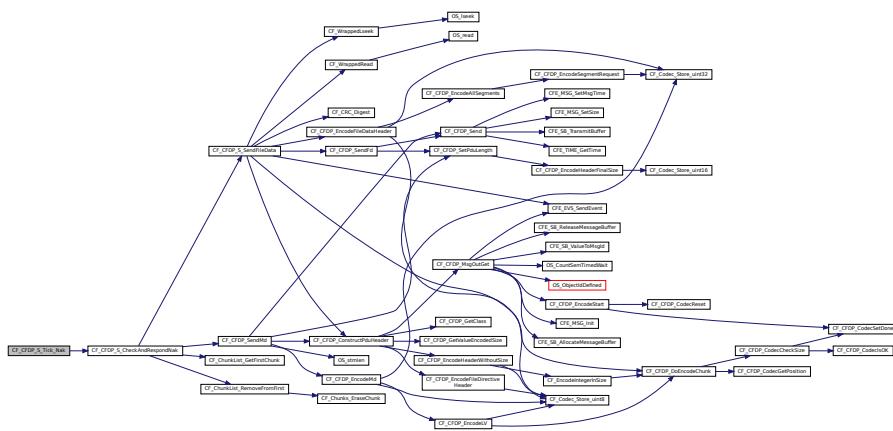
<i>txn</i>	Pointer to the transaction object
<i>cont</i>	Set to 1 if a NAK was generated

Definition at line 772 of file cf_cfdp.s.c.

References CF CFDP S CheckAndRespondNak().

Referenced by CF CFDP TickTransactions().

Here is the call graph for this function:



12.30 apps/cf/fsw/src/cf_cfdp_sbintf.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_sbintf.h"
#include <string.h>
#include "cf_assert.h"
```

Functions

- CF_Logical_PduBuffer_t * CF_CFDP_MsgOutGet (const CF_Transaction_t *txn, bool silent)

Obtain a message buffer to construct a PDU inside.

- void [CF_CFDP_Send](#) (uint8 chan_num, const [CF_Logical_PduBuffer_t](#) *ph)
Sends the current output buffer via the software bus.
- void [CF_CFDP_ReceiveMessage](#) ([CF_Channel_t](#) *chan)
Process received message on channel PDU input pipe.

12.30.1 Detailed Description

This is the interface to the CFE Software Bus for CF transmit/recv. Specifically this implements 3 functions used by the CFDP engine:

- [CF_CFDP_MsgOutGet\(\)](#) - gets a buffer prior to transmitting
- [CF_CFDP_Send\(\)](#) - sends the buffer from [CF_CFDP_MsgOutGet](#)
- [CF_CFDP_ReceiveMessage\(\)](#) - gets a received message

These functions were originally part of the CFDP engine itself but were split into a separate file, both to improve testability as well as to (potentially) allow interfaces to message/packet services other than the CFE software bus. However, Note that in this version, there is still a fair amount of CFDP engine mixed into these functions that would have to be isolated. Also note that the creation and deletion of SB pipes is not yet moved into this file.

12.30.2 Function Documentation

12.30.2.1 [CF_CFDP_MsgOutGet\(\)](#) [CF_Logical_PduBuffer_t](#)* [CF_CFDP_MsgOutGet](#) (

```
    const CF\_Transaction\_t * txn,
    bool silent )
```

Obtain a message buffer to construct a PDU inside.

Description

This performs the handshaking via semaphore with the consumer of the PDU. If the semaphore can be obtained, a software bus buffer is obtained and it is returned. If the semaphore is unavailable, then the current transaction is remembered for next engine cycle. If silent is true, then the event message is not printed in the case of no buffer available.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>silent</i>	If true, suppresses error events if no message can be allocated

Returns

Pointer to a [CF_Logical_PduBuffer_t](#) on success.

Return values

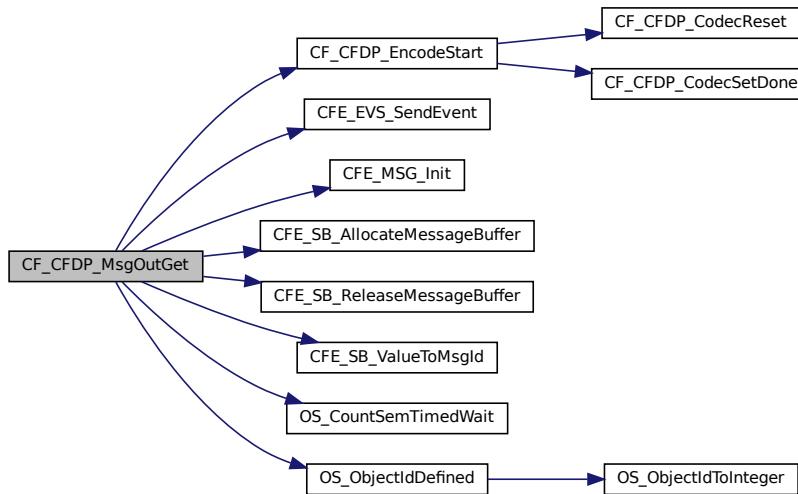
<i>NULL</i>	on error
-------------	----------

Definition at line 61 of file cf_cfdp_sbintf.c.

References CF_AppData, CF_CFDP_EncodeStart(), CF_CFDP_NO_MSG_ERR_EID, CF_MAX_PDU_SIZE, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_AllocateMessageBuffer(), CFE_SB_ReleaseMessageBuffer(), CFE_SB_ValueToMsgId(), CF_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_StateFlags::com, CF_AppData_t::config_table, CF_Channel::cur, CF_Output::encode, CF_AppData_t::engine, CF_Transaction::flags, CF_HkChannel_Data::frozen, CF_AppData_t::hk, CF_ChannelConfig::max_outgoing_messages_per_wakeup, CF_ChannelConfig::mid_output, CFE_SB_Msg::Msg, CF_Output::msg, OS_CountSemTimedWait(), OS_ObjectIdDefined(), OS_SUCCESS, CF_Engine::out, CF_Engine::outgoing_counter, CF_HkPacket::Payload, CF_Channel::sem_id, CF_Flags_Common::suspended, and CF_Output::tx_pdudata.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.30.2.2 CF_CFDP_ReceiveMessage()

```
void CF_CFDP_ReceiveMessage (
    CF_Channel_t * chan )
```

Process received message on channel PDU input pipe.

Assumptions, External Events, and Notes:

chan must be a member of the array within the CF_AppData global object

Parameters

chan	Channel to receive message on
------	-------------------------------

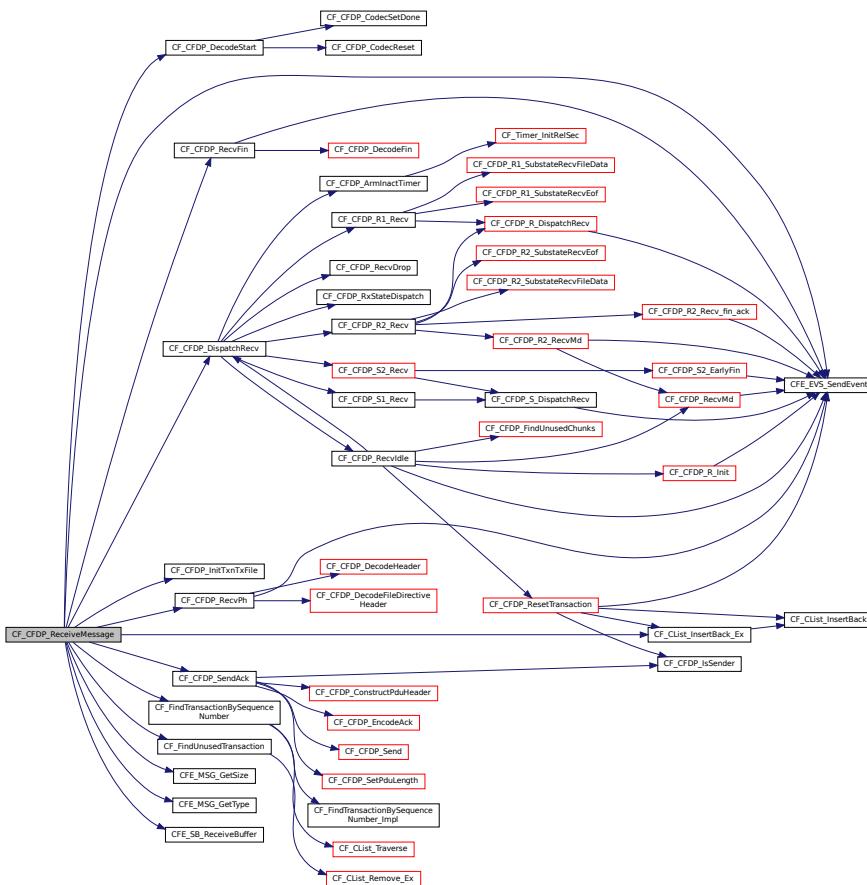
Definition at line 175 of file cf_cfdp_sbintf.c.

References CF_Logical_PduFin::cc, CF_AppData, CF_Assert, CF_CFDP_AckTxnStatus_UNRECOGNIZED, CF_CFDP_CLASS_2, CF_CFDP_DecodeStart(), CF_CFDP_DispatchRecv(), CF_CFDP_FileDirective_FIN, CF_CFDP_FinDeliveryCode_INCOMPLETE, CF_CFDP_FinFileStatus_DISCARDED, CF_CFDP_InitTxnTxFile(), CF_CFDP_INVALID_DST_ERR_EID, CF_CFDP_RecvFin(), CF_CFDP_RecvPh(), CF_CFDP_RX_DROPPED_ERR_EID, CF

```
_CFDP_SendAck(), CF_CList_InsertBack_Ex(), CF_Direction_RX, CF_FindTransactionBySequenceNumber(), CF_FindUnusedTransaction(), CF_MAX_SIMULTANEOUS_RX, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, CF_PERF_ID_PDURCVD, CF_QueueIdx_RX, CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_TxnState_IDLE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetSize(), CFE_MSG_GetType(), CFE_MSG_Type_Invalid, CFE_MSG_Type_Tlm, CFE_SB_POLL, CFE_SB_ReceiveBuffer(), CFE_SUCCESS, CF_ConfigTable::chan, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_Channel::cur, CF_RxS2_Data::dc, CF_Input::decode, CF_Logical_PduHeader::destination_eid, CF_History::dir, CF_Logical_PduFileDirectiveHeader::directive_code, CF_AppData_t::engine, CF_Logical_PduBuffer::fdirective, CF_Logical_IntHeader::fin, CF_Transaction::flags, CF_RxS2_Data::fs, CF_Transaction::history, CF_AppData_t::hk, CF_Engine::in, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CFE_SB_Msg::Msg, CF_HkPacket::Payload, CF_Logical_PduBuffer::pdu_header, CF_Channel::pipe, CF_Flags_Common::q_index, CF_HkChannel_Data::q_size, CF_RxState_Data::r2, CF_StateData::receive, CF_HkCounters::recv, CF_ChannelConfig::rx_max_messages_per_wakeup, CF_Input::rx_pdudata, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_HkRecv::spurious, CF_Transaction::state, and CF_Transaction::state_data.
```

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.30.2.3 CF_CFDP_Send()

```
    uint8 chan_num,  
    const CF_Logical_PduBuffer_t * ph )
```

Sends the current output buffer via the software bus.

Assumptions, External Events, and Notes:

The PDU in the output buffer is ready to transmit.

Parameters

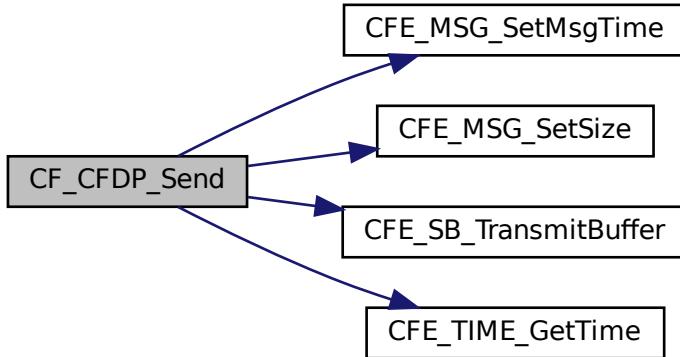
<i>chan_num</i>	Channel number for statistics/accounting purposes
<i>ph</i>	Pointer to PDU buffer to send

Definition at line 147 of file cf_cfdp_sbintf.c.

References CF_AppData, CF_ASSERT, CF_NUM_CHANNELS, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, CFE_MSG_SetMsgTime(), CFE_MSG_SetSize(), CFE_SB_TransmitBuffer(), CFE_TIME_GetTime(), CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduHeader::data_encoded_length, CF_AppData_t::engine, CF_Logical_PduHeader::header_encoded_length, CF_AppData_t::hk, CFE_SB_Msg::Msg, CF_Output::msg, CF_Engine::out, CF_HkPacket::Payload, CF_HkSent::pdu, CF_Logical_PduBuffer::pdu_header, and CF_HkCounters::sent.

Referenced by CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFd(), CF_CFDP_SendFin(), CF_CFDP_SendMd(), and CF_CFDP_SendNak().

Here is the call graph for this function:



12.31 apps/cf/fsw/src/cf_cfdp_sbintf.h File Reference

```
#include "cf_cfdp_types.h"
```

Data Structures

- struct [CF_PduCmdMsg](#)
PDU command encapsulation structure.
- struct [CF_PduTlmMsg](#)
PDU send encapsulation structure.

Typedefs

- `typedef struct CF_PduCmdMsg CF_PduCmdMsg_t`
PDU command encapsulation structure.
- `typedef struct CF_PduTlmMsg CF_PduTlmMsg_t`
PDU send encapsulation structure.

Functions

- `CF_Logical_PduBuffer_t * CF_CFDP_MsgOutGet (const CF_Transaction_t *txn, bool silent)`
Obtain a message buffer to construct a PDU inside.
- `void CF_CFDP_Send (uint8 chan_num, const CF_Logical_PduBuffer_t *ph)`
Sends the current output buffer via the software bus.
- `void CF_CFDP_ReceiveMessage (CF_Channel_t *chan)`
Process received message on channel PDU input pipe.

12.31.1 Detailed Description

This is the interface to the CFE Software Bus for PDU transmit/recv.

It may serve as a future point of abstraction to interface with message passing interfaces other than the CFE software bus, if necessary.

12.31.2 Typedef Documentation

12.31.2.1 `CF_PduCmdMsg_t` `typedef struct CF_PduCmdMsg CF_PduCmdMsg_t`

PDU command encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "command" encapsulation (even though these are not really commands).

Note

this is only the definition of the header. In reality all messages are larger than this, up to CF_MAX_PDU_SIZE.

12.31.2.2 `CF_PduTlmMsg_t` `typedef struct CF_PduTlmMsg CF_PduTlmMsg_t`

PDU send encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "telemetry" encapsulation (even though these are not really telemetry items).

Note

this is only the definition of the header. In reality all messages are larger than this, up to CF_MAX_PDU_SIZE.

12.31.3 Function Documentation

12.31.3.1 `CF_CFDP_MsgOutGet()` `CF_Logical_PduBuffer_t* CF_CFDP_MsgOutGet (` `const CF_Transaction_t * txn,` `bool silent)`

Obtain a message buffer to construct a PDU inside.

Description

This performs the handshaking via semaphore with the consumer of the PDU. If the semaphore can be obtained, a software bus buffer is obtained and it is returned. If the semaphore is unavailable, then the current transaction is remembered for next engine cycle. If silent is true, then the event message is not printed in the case of no buffer available.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>silent</i>	If true, suppresses error events if no message can be allocated

Returns

Pointer to a CF_Logical_PduBuffer_t on success.

Return values

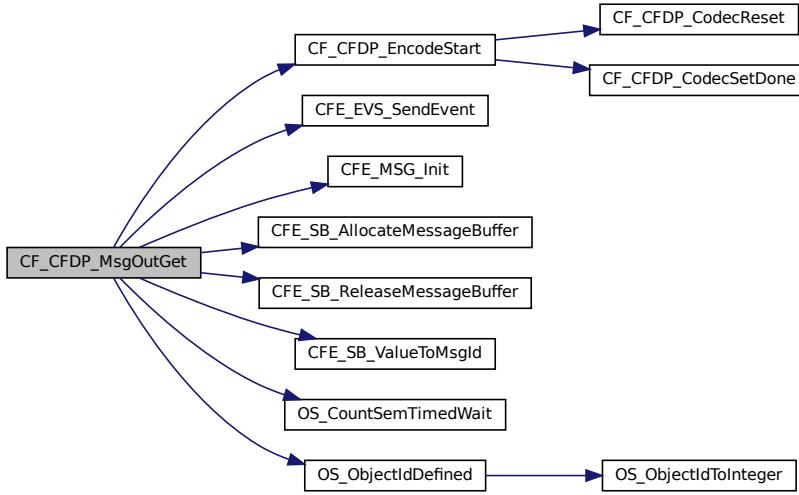
<i>NULL</i>	on error
-------------	----------

Definition at line 61 of file cf_cfdp_sbintf.c.

References CF_AppData, CF_CFDP_EncodeStart(), CF_CFDP_NO_MSG_ERR_EID, CF_MAX_PDU_SIZE, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_AllocateMessageBuffer(), CFE_SB_ReleaseMessageBuffer(), CFE_SB_ValueToMsgId(), CF_F_ConfigTable::chan, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_StateFlags::com, CF_AppData_t::config_table, CF_Channel::cur, CF_Output::encode, CF_AppData_t::engine, CF_Transaction::flags, CF_HkChannel_Data::frozen, CF_AppData_t::hk, CF_ChannelConfig::max_outgoing_messages_per_wakeup, CF_ChannelConfig::mid_output, CFE_SB_Msg::Msg, CF_Output::msg, OS_CountSemTimedWait(), OS_S_ObjectIdDefined(), OS_SUCCESS, CF_Engine::out, CF_Engine::outgoing_counter, CF_HkPacket::Payload, CF_Channel::sem_id, CF_Flags_Common::suspended, and CF_Output::tx_pdudata.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.31.3.2 CF_CFDP_ReceiveMessage() void CF_CFDP_ReceiveMessage (CF_Channel_t * chan)

Process received message on channel PDU input pipe.

Assumptions, External Events, and Notes:

chan must be a member of the array within the CF_AppData global object

Parameters

chan	Channel to receive message on
------	-------------------------------

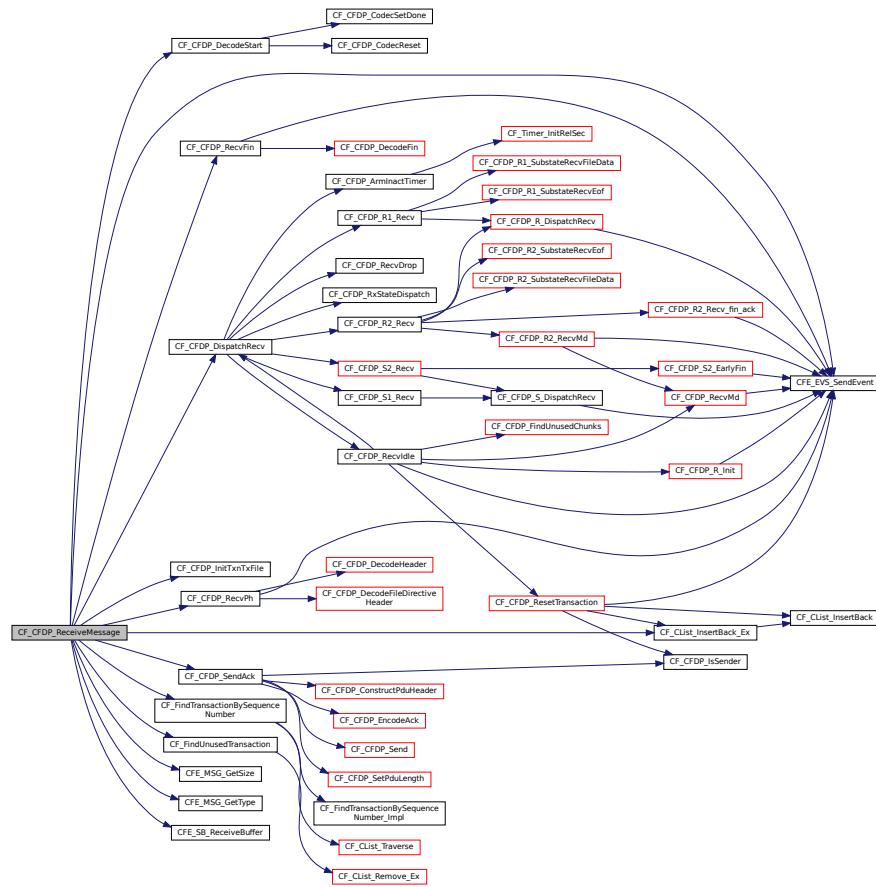
Definition at line 175 of file cf_cfdp_sbintf.c.

References CF_Logical_PduFin::cc, CF_AppData, CF_Assert, CF_CFDP_AckTxnStatus_UNRECOGNIZED, CF_CFDP_CLASS_2, CF_CFDP_DecodeStart(), CF_CFDP_DispatchRecv(), CF_CFDP_FileDirective_FIN, CF_CFDP_FinDeliveryCode_INCOMPLETE, CF_CFDP_FinFileStatus_DISCARDED, CF_CFDP_InitTxnTxFile(), CF_CFDP_INVALID_DST_ERR_EID, CF_CFDP_RecvFin(), CF_CFDP_RecvPh(), CF_CFDP_RX_DROPPED_ERR_EID, CF_CFDP_SendAck(), CF_CList_InsertBack_Ex(), CF_Direction_RX, CF_FindTransactionBySequenceNumber(), CF_FindUnusedTransaction(), CF_MAX_SIMULTANEOUS_RX, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYT_ES, CF_PERF_ID_PDURCVD, CF_QueueIdx_RX, CF_SEND_PDU_NO_BUF_AVAIL_ERROR, CF_TxnState_IDLE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MS_G.GetSize(), CFE_MSG_GetType(), CFE_MSG_Type_Invalid, CFE_MSG_Type_Tlm, CFE_SB_POLL, CFE_SB_ReceiveBuffer(), CFE_SUCCESS, CF_ConfigTable::chan, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::config_table, CF_HkChannel_Data::counters, CF_Channel::cur, CF_RxS2_Data::dc, CF_Input::decode, CF_Logical_PduHeader::destination_eid, CF_History::dir, CF_Logical_PduFileDirectiveHeader::directive_code, CF_AppData_t::engine, CF_Logical_PduBuffer::fdirective, CF_Logical_InthHeader::fin, CF_Transaction::flags, CF_RxS2_Data::fs, CF_Transaction::history, CF_AppData_t::hk, CF_Engine::in, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CFE_SB_Msg::Msg, CF_HkPacket::

::Payload, CF_Logical_PduBuffer::pdu_header, CF_Channel::pipe, CF_Flags_Common::q_index, CF_HkChannel<_Data::q_size, CF_RxState_Data::r2, CF_StateData::receive, CF_HkCounters::recv, CF_ChannelConfig::rx_max<messages_per_wakeup, CF_Input::rx_pdudata, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader<::source_eid, CF_HkRecv::spurious, CF_Transaction::state, and CF_Transaction::state_data.

Referenced by CF_CFDP_CycleEngine().

Here is the call graph for this function:



12.31.3.3 CF_CFDP_Send() void CF_CFDP_Send (
 uint8 chan_num,
 const CF_Logical_PduBuffer_t * ph)

Sends the current output buffer via the software bus.

Assumptions, External Events, and Notes:

The PDU in the output buffer is ready to transmit.

Parameters

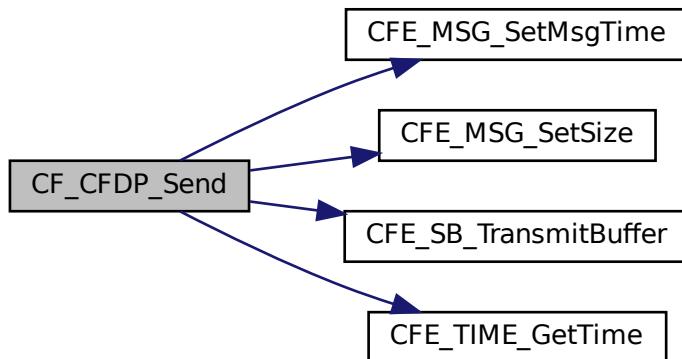
chan_num	Channel number for statistics/accounting purposes
ph	Pointer to PDU buffer to send

Definition at line 147 of file cf_cfdp_sbintf.c.

References CF_AppData, CF_Assert, CF_NUM_CHANNELS, CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, CFE_MSG_SetMsgTime(), CFE_MSG_SetSize(), CFE_SB_TransmitBuffer(), CFE_TIME_GetTime(), CF_HkPacket_Payload::channel_hk, CF_HkChannel_Data::counters, CF_Logical_PduHeader::data_encoded_length, CF_AppData_t::engine, CF_Logical_PduHeader::header_encoded_length, CF_AppData_t::hk, CFE_SB_Msg::Msg, CF_Output::msg, CF_Engine::out, CF_HkPacket::Payload, CF_HkSent::pdu, CF_Logical_PduBuffer::pdu_header, and CF_HkCounters::sent.

Referenced by CF_CFDP_SendAck(), CF_CFDP_SendEof(), CF_CFDP_SendFd(), CF_CFDP_SendFin(), CF_CFD_P_SendMd(), and CF_CFDP_SendNak().

Here is the call graph for this function:



12.32 apps/cf/fsw/src(cf_cfdp_types.h File Reference

```
#include "common_types.h"
#include "cf_cfdp_pdu.h"
#include "cf_extern_typedefs.h"
#include "cf_platform_cfg.h"
#include "cf_msg.h"
#include "cf_clist.h"
#include "cf_chunk.h"
#include "cf_timer.h"
#include "cf_crc.h"
#include "cf_codec.h"
```

Data Structures

- struct [CF_History](#)
CF History entry.
- struct [CF_ChunkWrapper](#)
Wrapper around a CF_ChunkList_t object.
- struct [CF_Playback](#)
CF Playback entry.
- struct [CF_Poll](#)

- *CF Poll entry.*
- struct [CF_TxS2_Data](#)
Data specific to a class 2 send file transaction.
- struct [CF_TxState_Data](#)
Data specific to a send file transaction.
- struct [CF_RxS2_Data](#)
Data specific to a class 2 receive file transaction.
- struct [CF_RxState_Data](#)
Data specific to a receive file transaction.
- struct [CF_Flags_Common](#)
Data that applies to all types of transactions.
- struct [CF_Flags_Rx](#)
Flags that apply to receive transactions.
- struct [CF_Flags_Tx](#)
Flags that apply to send transactions.
- union [CF_StateFlags](#)
Summary of all possible transaction flags (tx and rx)
- union [CF_StateData](#)
Summary of all possible transaction state information (tx and rx)
- struct [CF_Transaction](#)
Transaction state object.
- struct [CF_Channel](#)
Channel state object.
- struct [CF_Output](#)
CF engine output state.
- struct [CF_Input](#)
CF engine input state.
- struct [CF_Engine](#)
An engine represents a pairing to a local EID.

Macros

- #define [CF_NUM_TRANSACTIONS_PER_CHANNEL](#)
Maximum possible number of transactions that may exist on a single CF channel.
- #define [CF_NUM_TRANSACTIONS \(CF_NUM_CHANNELS * CF_NUM_TRANSACTIONS_PER_CHANNEL\)](#)
Maximum possible number of transactions that may exist in the CF application.
- #define [CF_NUM_HISTORIES \(CF_NUM_CHANNELS * CF_NUM_HISTORIES_PER_CHANNEL\)](#)
Maximum possible number of history entries that may exist in the CF application.
- #define [CF_NUM_CHUNKS_ALL_CHANNELS \(CF_TOTAL_CHUNKS * CF_NUM_TRANSACTIONS_PER_CHANNEL\)](#)
Maximum possible number of chunk entries that may exist in the CF application.

Typedefs

- typedef struct [CF_History](#) [CF_History_t](#)
CF History entry.
- typedef struct [CF_ChunkWrapper](#) [CF_ChunkWrapper_t](#)
Wrapper around a CF_ChunkList_t object.
- typedef struct [CF_Playback](#) [CF_Playback_t](#)

- CF Playback entry.*
- **typedef struct CF_Poll CF_Poll_t**
CF Poll entry.
 - **typedef struct CF_TxS2_Data CF_TxS2_Data_t**
Data specific to a class 2 send file transaction.
 - **typedef struct CF_TxState_Data CF_TxState_Data_t**
Data specific to a send file transaction.
 - **typedef struct CF_RxS2_Data CF_RxS2_Data_t**
Data specific to a class 2 receive file transaction.
 - **typedef struct CF_RxState_Data CF_RxState_Data_t**
Data specific to a receive file transaction.
 - **typedef struct CF_Flags_Common CF_Flags_Common_t**
Data that applies to all types of transactions.
 - **typedef struct CF_Flags_Rx CF_Flags_Rx_t**
Flags that apply to receive transactions.
 - **typedef struct CF_Flags_Tx CF_Flags_Tx_t**
Flags that apply to send transactions.
 - **typedef union CF_StateFlags CF_StateFlags_t**
Summary of all possible transaction flags (tx and rx)
 - **typedef union CF_StateData CF_StateData_t**
Summary of all possible transaction state information (tx and rx)
 - **typedef struct CF_Transaction CF_Transaction_t**
Transaction state object.
 - **typedef struct CF_Channel CF_Channel_t**
Channel state object.
 - **typedef struct CF_Output CF_Output_t**
CF engine output state.
 - **typedef struct CF_Input CF_Input_t**
CF engine input state.
 - **typedef struct CF_Engine CF_Engine_t**
An engine represents a pairing to a local EID.

Enumerations

- **enum CF_TxnState_t {**
 CF_TxnState_IDLE = 0, CF_TxnState_R1 = 1, CF_TxnState_S1 = 2, CF_TxnState_R2 = 3,
 CF_TxnState_S2 = 4, CF_TxnState_DROP = 5, CF_TxnState_INVALID = 6 }
High-level state of a transaction.
- **enum CF_TxSubState_t {**
 CF_TxSubState_METADATA = 0, CF_TxSubState_FILEDATA = 1, CF_TxSubState_EOF = 2, CF_TxSubState_WAIT_FOR_EOF_A
 = 3,
 CF_TxSubState_WAIT_FOR_FIN = 4, CF_TxSubState_SEND_FIN_ACK = 5, CF_TxSubState_NUM_STATES
 = 6 }
Sub-state of a send file transaction.
- **enum CF_RxSubState_t {**
 CF_RxSubState_FILEDATA = 0, CF_RxSubState_EOF = 1, CF_RxSubState_WAIT_FOR_FIN_ACK
 = 2, CF_RxSubState_NUM_STATES = 3 }
Sub-state of a receive file transaction.
- **enum CF_Direction_t {**
 CF_Direction_RX = 0, CF_Direction_TX = 1, CF_Direction_NUM = 2 }

Direction identifier.

- enum CF_TxnStatus_t {
 CF_TxnStatus_UNDEFINED = -1, CF_TxnStatus_NO_ERROR = CF_CFDP_ConditionCode_NO_ERROR,
 CF_TxnStatus_POS_ACK_LIMIT_REACHED = CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED,
 CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED = CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED,
 CF_TxnStatus_INVALID_TRANSMISSION_MODE = CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE,
 CF_TxnStatus_FILESTORE_REJECTION = CF_CFDP_ConditionCode_FILESTORE_REJECTION,
 CF_TxnStatus_FILE_CHECKSUM_FAILURE = CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE,
 CF_TxnStatus_FILE_SIZE_ERROR = CF_CFDP_ConditionCode_FILE_SIZE_ERROR,
 CF_TxnStatus_NAK_LIMIT_REACHED = CF_CFDP_ConditionCode_NAK_LIMIT_REACHED,
 CF_TxnStatus_INACTIVITY_DETECTED = CF_CFDP_ConditionCode_INACTIVITY_DETECTED,
 CF_TxnStatus_INVALID_FILE_STRUCTURE = CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE,
 CF_TxnStatus_CHECK_LIMIT_REACHED = CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED,
 CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE = CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE,
 CF_TxnStatus_SUSPEND_REQUEST_RECEIVED = CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED,
 CF_TxnStatus_CANCEL_REQUEST_RECEIVED = CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED,
 CF_TxnStatus_PROTOCOL_ERROR = 16,
 CF_TxnStatus_ACK_LIMIT_NO_FIN = 17, CF_TxnStatus_ACK_LIMIT_NO_EOF = 18,
 CF_TxnStatus_NAK_RESPONSE_ERROR = 19, CF_TxnStatus_SEND_EOF_FAILURE = 20,
 CF_TxnStatus_EARLY_FIN = 21, CF_TxnStatus_MAX = 22 }

Values for Transaction Status code.

- enum CF_TickType_t { CF_TickType_RX, CF_TickType_TXW_NORM, CF_TickType_TXW_NAK, CF_TickType_NUM_TYPES }

Identifies the type of timer tick being processed.

12.32.1 Detailed Description

Macros and data types used across the CF application

Note

Functions should not be declared in this file. This should be limited to shared macros and data types only. For unit testing, functions should be declared only in a header file with the same name as the C file that defines that function.

12.32.2 Macro Definition Documentation

12.32.2.1 CF_NUM_CHUNKS_ALL_CHANNELS `#define CF_NUM_CHUNKS_ALL_CHANNELS (CF_TOTAL_CHUNKS * CF_NUM_TRANSACTIONS_PER_CHANNEL)`

Maximum possible number of chunk entries that may exist in the CF application.

Definition at line 66 of file cf_cfdp_types.h.

12.32.2.2 CF_NUM_HISTORIES `#define CF_NUM_HISTORIES (CF_NUM_CHANNELS * CF_NUM_HISTORIES_PER_CHANNEL)`

Maximum possible number of history entries that may exist in the CF application.

Definition at line 61 of file cf_cfdp_types.h.

12.32.2.3 CF_NUM_TRANSACTIONS `#define CF_NUM_TRANSACTIONS (CF_NUM_CHANNELS * CF_NUM_TRANSACTIONS_PER_CHANNEL)`

Maximum possible number of transactions that may exist in the CF application.

Definition at line 56 of file cf_cfdp_types.h.

12.32.2.4 CF_NUM_TRANSACTIONS_PER_CHANNEL #define CF_NUM_TRANSACTIONS_PER_CHANNEL**Value:**

```
(CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN + CF_MAX_SIMULTANEOUS_RX +  
((CF_MAX_POLLING_DIR_PER_CHAN + CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN) * \  
CF_NUM_TRANSACTIONS_PER_PLAYBACK))
```

Maximum possible number of transactions that may exist on a single CF channel.

Definition at line 48 of file cf_cfdp_types.h.

12.32.3 Typedef Documentation

12.32.3.1 CF_Channel_t typedef struct CF_Channel CF_Channel_t

Channel state object.

This keeps the state of CF channels

Each CF channel has a separate transaction list, PDU throttle, playback, and poll state, as well as separate addresses on the underlying message transport (e.g. SB).

12.32.3.2 CF_ChunkWrapper_t typedef struct CF_ChunkWrapper CF_ChunkWrapper_t

Wrapper around a CF_ChunkList_t object.

This allows a CF_ChunkList_t to be stored within a CList data storage structure

12.32.3.3 CF_Engine_t typedef struct CF_Engine CF_Engine_t

An engine represents a pairing to a local EID.

Each engine can have at most CF_MAX_SIMULTANEOUS_TRANSACTIONS

12.32.3.4 CF_Flags_Common_t typedef struct CF_Flags_Common CF_Flags_Common_t

Data that applies to all types of transactions.

12.32.3.5 CF_Flags_Rx_t typedef struct CF_Flags_Rx CF_Flags_Rx_t

Flags that apply to receive transactions.

12.32.3.6 CF_Flags_Tx_t typedef struct CF_Flags_Tx CF_Flags_Tx_t

Flags that apply to send transactions.

12.32.3.7 CF_History_t typedef struct CF_History CF_History_t

CF History entry.

Records CF app operations for future reference

12.32.3.8 CF_Input_t typedef struct CF_Input CF_Input_t

CF engine input state.

Keeps the state of the current input PDU in the CF engine

12.32.3.9 CF_Output_t typedef struct CF_Output CF_Output_t

CF engine output state.

Keeps the state of the current output PDU in the CF engine

12.32.3.10 CF_Playback_t typedef struct CF_Playback CF_Playback_t

CF Playback entry.

Keeps the state of CF playback requests

12.32.3.11 CF_Poll_t `typedef struct CF_Poll CF_Poll_t`
CF Poll entry.
Keeps the state of CF directory polling

12.32.3.12 CF_RxS2_Data_t `typedef struct CF_RxS2_Data CF_RxS2_Data_t`
Data specific to a class 2 receive file transaction.

12.32.3.13 CF_RxState_Data_t `typedef struct CF_RxState_Data CF_RxState_Data_t`
Data specific to a receive file transaction.

12.32.3.14 CF_StateData_t `typedef union CF_StateData CF_StateData_t`
Summary of all possible transaction state information (tx and rx)

12.32.3.15 CF_StateFlags_t `typedef union CF_StateFlags CF_StateFlags_t`
Summary of all possible transaction flags (tx and rx)

12.32.3.16 CF_Transaction_t `typedef struct CF_Transaction CF_Transaction_t`
Transaction state object.
This keeps the state of CF file transactions

12.32.3.17 CF_TxS2_Data_t `typedef struct CF_TxS2_Data CF_TxS2_Data_t`
Data specific to a class 2 send file transaction.

12.32.3.18 CF_TxState_Data_t `typedef struct CF_TxState_Data CF_TxState_Data_t`
Data specific to a send file transaction.

12.32.4 Enumeration Type Documentation

12.32.4.1 CF_Direction_t `enum CF_Direction_t`
Direction identifier.
Differentiates between send and receive history entries

Enumerator

CF_Direction_RX	
CF_Direction_TX	
CF_Direction_NUM	

Definition at line 112 of file cf_cfdp_types.h.

12.32.4.2 CF_RxSubState_t `enum CF_RxSubState_t`
Sub-state of a receive file transaction.

Enumerator

CF_RxSubState_FILEDATA	
CF_RxSubState_EOF	
CF_RxSubState_WAIT_FOR_FIN_ACK	
CF_RxSubState_NUM_STATES	

Definition at line 99 of file cf_cfdp_types.h.

12.32.4.3 CF_TickType_t enum CF_TickType_t

Identifies the type of timer tick being processed.

Enumerator

CF_TickType_RX	
CF_TickType_TXW_NORM	
CF_TickType_TXW_NAK	
CF_TickType_NUM_TYPES	

Definition at line 375 of file cf_cfdp_types.h.

12.32.4.4 CF_TxnState_t enum CF_TxnState_t

High-level state of a transaction.

Enumerator

CF_TxnState_IDLE	State assigned to a newly allocated transaction object.
CF_TxnState_R1	Receive file as class 1.
CF_TxnState_S1	Send file as class 1.
CF_TxnState_R2	Receive file as class 2.
CF_TxnState_S2	Send file as class 2.
CF_TxnState_DROP	State where all PDUs are dropped.
CF_TxnState_INVALID	Marker value for the highest possible state number.

Definition at line 71 of file cf_cfdp_types.h.

12.32.4.5 CF_TxnStatus_t enum CF_TxnStatus_t

Values for Transaction Status code.

This enum defines the possible values representing the result of a transaction. This is a superset of the condition codes defined in CCSDS book 727 (condition codes) but with additional values for local conditions that the blue book does not have, such as protocol/state machine or decoding errors.

The values here are designed to not overlap with the condition codes defined in the blue book, but can be translated to one of those codes for the purposes of FIN/ACK/EOF PDUs.

Enumerator

CF_TxnStatus_UNDEFINED	The undefined status is a placeholder for new transactions before a value is set.
------------------------	---

Enumerator

CF_TxnStatus_NO_ERROR
CF_TxnStatus_POS_ACK_LIMIT_REACHED
CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED
CF_TxnStatus_INVALID_TRANSMISSION_MODE
CF_TxnStatus_FILESTORE_REJECTION
CF_TxnStatus_FILE_CHECKSUM_FAILURE
CF_TxnStatus_FILE_SIZE_ERROR
CF_TxnStatus_NAK_LIMIT_REACHED
CF_TxnStatus_INACTIVITY_DETECTED
CF_TxnStatus_INVALID_FILE_STRUCTURE
CF_TxnStatus_CHECK_LIMIT_REACHED
CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE
CF_TxnStatus_SUSPEND_REQUEST_RECEIVED
CF_TxnStatus_CANCEL_REQUEST_RECEIVED
CF_TxnStatus_PROTOCOL_ERROR
CF_TxnStatus_ACK_LIMIT_NO_FIN
CF_TxnStatus_ACK_LIMIT_NO_EOF
CF_TxnStatus_NAK_RESPONSE_ERROR
CF_TxnStatus_SEND_EOF_FAILURE
CF_TxnStatus_EARLY_FIN
CF_TxnStatus_MAX

Definition at line 132 of file cf_cfdp_types.h.

12.32.4.6 CF_TxSubState_t enum CF_TxSubState_t

Sub-state of a send file transaction.

Enumerator

CF_TxSubState_METADATA
CF_TxSubState_FILEDATA
CF_TxSubState_EOF
CF_TxSubState_WAIT_FOR_EOF_ACK
CF_TxSubState_WAIT_FOR_FIN
CF_TxSubState_SEND_FIN_ACK
CF_TxSubState_NUM_STATES

Definition at line 85 of file cf_cfdp_types.h.

12.33 apps/cf/fsw/src(cf_chunk.c File Reference

```
#include <string.h>
#include "cf_verify.h"
#include "cf_assert.h"
#include "cf_chunk.h"
```

Functions

- void `CF_Chunks_EraseRange (CF_ChunkList_t *chunks, CF_ChunkIdx_t start, CF_ChunkIdx_t end)`
Erase a range of chunks.
- void `CF_Chunks_EraseChunk (CF_ChunkList_t *chunks, CF_ChunkIdx_t erase_index)`
Erase a single chunk.
- void `CF_Chunks_InsertChunk (CF_ChunkList_t *chunks, CF_ChunkIdx_t index_before, const CF_Chunk_t *chunk)`
Insert a chunk before index_before.
- `CF_ChunkIdx_t CF_Chunks_FindInsertPosition (CF_ChunkList_t *chunks, const CF_Chunk_t *chunk)`
Finds where a chunk should be inserted in the chunks.
- int `CF_Chunks_CombinePrevious (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`
Possibly combines the given chunk with the previous chunk.
- `CFE_Status_t CF_Chunks_CombineNext (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`
Possibly combines the given chunk with the next chunk.
- `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (const CF_ChunkList_t *chunks)`
Finds the smallest size out of all chunks.
- void `CF_Chunks_Insert (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`
Insert a chunk.
- void `CF_ChunkListAdd (CF_ChunkList_t *chunks, CF_ChunkOffset_t offset, CF_ChunkSize_t size)`
Public function to add a chunk.
- void `CF_ChunkList_RemoveFromFirst (CF_ChunkList_t *chunks, CF_ChunkSize_t size)`
Public function to remove some amount of size from the first chunk.
- const `CF_Chunk_t * CF_ChunkList_GetFirstChunk (const CF_ChunkList_t *chunks)`
Public function to get the entire first chunk from the list.
- void `CF_ChunkListInit (CF_ChunkList_t *chunks, CF_ChunkIdx_t max_chunks, CF_Chunk_t *chunks_mem)`
Initialize a CF_ChunkList_t structure.
- void `CF_ChunkListReset (CF_ChunkList_t *chunks)`
Resets a chunks structure.
- uint32 `CF_ChunkList_ComputeGaps (const CF_ChunkList_t *chunks, CF_ChunkIdx_t max_gaps, CF_ChunkSize_t total, CF_ChunkOffset_t start, CF_ChunkList_ComputeGapFn_t compute_gap_fn, void *opaque)`
Compute gaps between chunks, and call a callback for each.

12.33.1 Detailed Description

The CF Application chunks (sparse gap tracking) logic file

This class handles the complexity of sparse gap tracking so that the CFDP engine doesn't need to worry about it.

Information is given to the class and when needed calculations are made internally to help the engine build NAK packets.

Received NAK segment requests are stored in this class as well and used for re-transmit processing.

This is intended to be mostly a generic purpose class used by CF.

12.33.2 Function Documentation

```
12.33.2.1 CF_ChunkList_ComputeGaps() uint32 CF_ChunkList_ComputeGaps (
    const CF_ChunkList_t * chunks,
    CF_ChunkIdx_t max_gaps,
    CF_ChunkSize_t total,
    CF_ChunkOffset_t start,
    CF_ChunkList_ComputeGapFn_t compute_gap_fn,
    void * opaque )
```

Compute gaps between chunks, and call a callback for each.

Description

This function walks over all chunks and computes the gaps between. It can exit early if the calculated gap start is larger than the desired total.

Assumptions, External Events, and Notes:

chunks must not be NULL. compute_gap_fn is a valid function address.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>max_gaps</i>	Maximum number of gaps to compute
<i>total</i>	Size of the entire file
<i>start</i>	Beginning offset for gap computation
<i>compute_gap_fn</i>	Callback function to be invoked for each gap
<i>opaque</i>	Opaque pointer to pass through to callback function

Returns

The number of computed gaps.

Definition at line 362 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_ChunkList::count, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_CFDP_R2_Complete(), and CF_CFDP_R_SubstateSendNak().

```
12.33.2.2 CF_ChunkList_GetFirstChunk() const CF_Chunk_t* CF_ChunkList_GetFirstChunk (
    const CF_ChunkList_t * chunks )
```

Public function to get the entire first chunk from the list.

This returns the first chunk from the list, or NULL if the list is empty.

Note

The chunk remains on the list - this call does not consume or remove the chunk from the list.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Returns

Pointer to first chunk from the CF_ChunkList_t object

Return values

<code>NULL</code>	if the list was empty
-------------------	-----------------------

Definition at line 325 of file cf_chunk.c.

References CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_CFDP_S_CheckAndRespondNak().

12.33.2.3 CF_ChunkList_RemoveFromFirst() `void CF_ChunkList_RemoveFromFirst (`

```
    CF_ChunkList_t * chunks,
    CF_ChunkSize_t size )
```

Public function to remove some amount of size from the first chunk.

Description

This may remove the chunk entirely. This function is to satisfy the use-case where data is retrieved from the structure in-order and once consumed should be removed.

Assumptions, External Events, and Notes:

chunks must not be NULL and list must not be empty

Note

Good computer science would have a generic remove function, but that's much more complex than we need for the use case. We aren't trying to make chunks a general purpose reusable module, so just take the simple case that we need.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>size</code>	Size to remove

Definition at line 299 of file cf_chunk.c.

References CF_Chunks_EraseChunk(), CF_ChunkList::chunks, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_CFDP_S_CheckAndRespondNak().

Here is the call graph for this function:

**12.33.2.4 CF_ChunkListAdd()** `void CF_ChunkListAdd (`

```
    CF_ChunkList_t * chunks,
    CF_ChunkOffset_t offset,
    CF_ChunkSize_t size )
```

Public function to add a chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

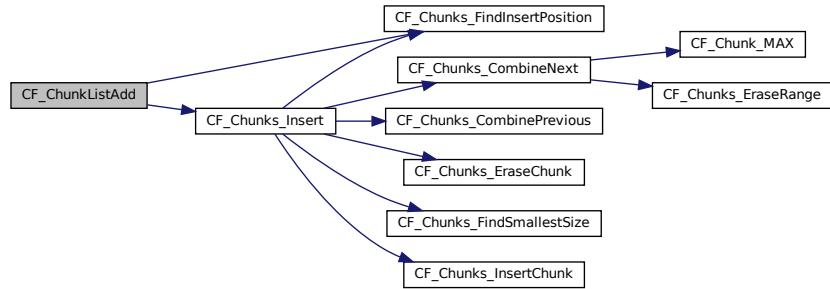
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>offset</i>	Offset of chunk to add
<i>size</i>	Size of chunk to add

Definition at line 280 of file cf_chunk.c.

References CF_ASSERT, CF_Chunks_FindInsertPosition(), and CF_Chunks_Insert().

Referenced by CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_S2_Nak().

Here is the call graph for this function:



12.33.2.5 CF_ChunkListInit() void CF_ChunkListInit (

```

CF_ChunkList_t * chunks,
CF_ChunkIdx_t max_chunks,
CF_Chunk_t * chunks_mem )
  
```

Initialize a CF_ChunkList_t structure.

Assumptions, External Events, and Notes:

chunks must not be NULL. chunks_mem must not be NULL.

Parameters

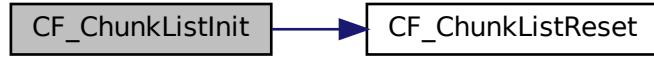
<i>chunks</i>	Pointer to CF_ChunkList_t object to initialize
<i>max_chunks</i>	Maximum number of entries in the chunks_mem array
<i>chunks_mem</i>	Array of CF_Chunk_t objects with length of max_chunks

Definition at line 336 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkListReset(), CF_ChunkList::chunks, and CF_ChunkList::max_chunks.

Referenced by CF_CFDP_InitEngine().

Here is the call graph for this function:



12.33.2.6 CF_ChunkListReset() `void CF_ChunkListReset (`
 `CF_ChunkList_t * chunks)`

Resets a chunks structure.

All chunks are removed from the list, but the max_chunks and chunk memory pointers are retained. This returns the chunk list to the same state as it was after the initial call to [CF_ChunkListInit\(\)](#).

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
---------------------	----------------------------------

Definition at line 350 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_ChunkList::max_chunks.

Referenced by CF_ChunkListInit().

12.33.2.7 CF_Chunks_CombineNext() `CFE_Status_t CF_Chunks_CombineNext (`
 `CF_ChunkList_t * chunks,`
 `CF_ChunkIdx_t i,`
 `const CF_Chunk_t * chunk)`

Possibly combines the given chunk with the next chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>i</code>	Index of chunk to combine
<code>chunk</code>	Chunk data to combine

Returns

boolean code indicating if chunks were combined

Return values

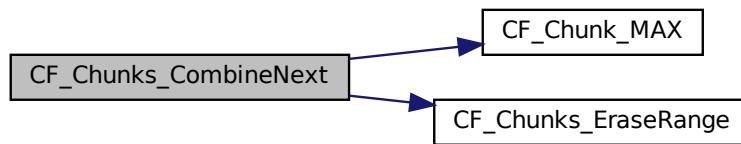
<i>1</i>	if combined with another chunk
<i>0</i>	if not combined

Definition at line 170 of file cf_chunk.c.

References CF_ASSERT, CF_Chunk_MAX(), CF_Chunks_EraseRange(), CF_ChunkList::chunks, CF_ChunkList::count, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

Here is the call graph for this function:



12.33.2.8 CF_Chunks_CombinePrevious() int CF_Chunks_CombinePrevious (
CF_ChunkList_t * *chunks*,
CF_ChunkIdx_t *i*,
const CF_Chunk_t * *chunk*)

Possibly combines the given chunk with the previous chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL, *chunk* must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Index of chunk to combine
<i>chunk</i>	Chunk data to combine

Returns

boolean code indicating if chunks were combined

Return values

<i>1</i>	if combined with another chunk
<i>0</i>	if not combined

Definition at line 133 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

12.33.2.9 CF_Chunks_EraseChunk() void CF_Chunks_EraseChunk (

```
CF_ChunkList_t * chunks,
CF_ChunkIdx_t erase_index )
```

Erase a single chunk.

Note

This changes the chunk IDs of all chunks that follow items in the list after the `erase_index` will be shifted/moved to close the gap.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>erase_index</code>	chunk ID to erase

Definition at line 63 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_ChunkList_RemoveFromFirst(), and CF_Chunks_Insert().

12.33.2.10 CF_Chunks_EraseRange() void CF_Chunks_EraseRange (

```
CF_ChunkList_t * chunks,
CF_ChunkIdx_t start,
CF_ChunkIdx_t end )
```

Erase a range of chunks.

Items in the list starting at the end index will be shifted/moved to close the gap. If `end <= start` nothing will be done (OK to pass matching start/end as a no-op) If `end == list size`, list from start on will be erased (nothing moved)

Example: list = {a, b, c, d, e} EraseRange index start 1, end 3 list = {a, d, e}

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>start</code>	Starting chunk ID to erase (inclusive)
<code>end</code>	Ending chunk ID (exclusive, this chunk will not be erased)

Definition at line 45 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_Chunks_CombineNext().

12.33.2.11 CF_Chunks_FindInsertPosition() `CF_ChunkIdx_t CF_Chunks_FindInsertPosition (`
 `CF_ChunkList_t * chunks,`
 `const CF_Chunk_t * chunk)`

Finds where a chunk should be inserted in the chunks.

Description

This is a C version of std::lower_bound from C++ algorithms.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>chunk</code>	Chunk data to insert

Returns

an index to the first chunk that is greater than or equal to the requested's offset.

Definition at line 101 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_Chunk::offset.

Referenced by CF_ChunkListAdd(), and CF_Chunks_Insert().

12.33.2.12 CF_Chunks_FindSmallestSize() `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (`
 `const CF_ChunkList_t * chunks)`

Finds the smallest size out of all chunks.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
---------------------	----------------------------------

Returns

The chunk index with the smallest size.

Return values

<code>0</code>	if the chunk list is empty
----------------	----------------------------

Definition at line 214 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

```
12.33.2.13 CF_Chunks_Insert() void CF_Chunks_Insert (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t i,
    const CF_Chunk_t * chunk )
```

Insert a chunk.

Description

Inserts the chunk at the specified location. May combine with an existing chunk if contiguous.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

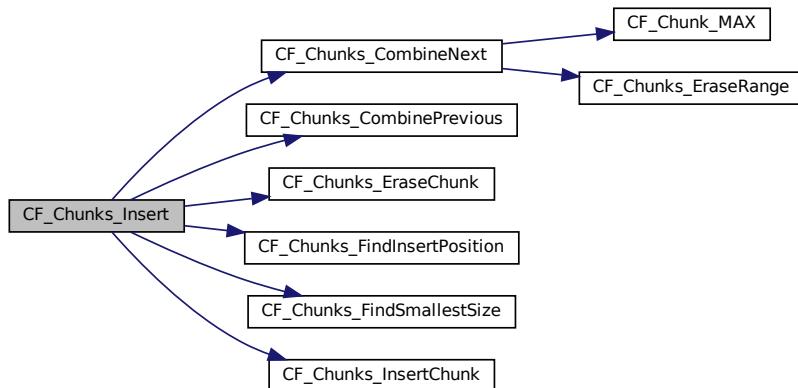
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Position to insert chunk at
<i>chunk</i>	Chunk data to insert

Definition at line 236 of file cf_chunk.c.

References CF_Chunks_CombineNext(), CF_Chunks_CombinePrevious(), CF_Chunks_EraseChunk(), CF_Chunks_FindInsertPosition(), CF_Chunks_FindSmallestSize(), CF_Chunks_InsertChunk(), CF_ChunkList::chunks, CF_ChunkList::count, CF_ChunkList::max_chunks, and CF_Chunk::size.

Referenced by CF_ChunkListAdd().

Here is the call graph for this function:



```
12.33.2.14 CF_Chunks_InsertChunk() void CF_Chunks_InsertChunk (
```

```
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t index_before,
    const CF_Chunk_t * chunk )
```

Insert a chunk before index_before.

Note

This changes the chunk IDs of all chunks that follow items in the list after the index_before will be shifted/moved to open a gap.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>index_before</i>	position to insert at - this becomes the ID of the inserted chunk
<i>chunk</i>	Chunk data to insert (copied)

Definition at line 80 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_ChunkList::count, and CF_ChunkList::max_chunks.
Referenced by CF_Chunks_Insert().

12.34 apps/cf/fsw/src(cf_chunk.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct [CF_Chunk](#)
Pairs an offset with a size to identify a specific piece of a file.
- struct [CF_ChunkList](#)
A list of CF_Chunk_t pairs.

TypeDefs

- typedef uint32 [CF_ChunkIdx_t](#)
- typedef uint32 [CF_ChunkOffset_t](#)
- typedef uint32 [CF_ChunkSize_t](#)
- typedef struct [CF_Chunk](#) [CF_Chunk_t](#)
Pairs an offset with a size to identify a specific piece of a file.
- typedef struct [CF_ChunkList](#) [CF_ChunkList_t](#)
A list of CF_Chunk_t pairs.
- typedef void(* [CF_ChunkList_ComputeGapFn_t](#)) (const [CF_ChunkList_t](#) *cs, const [CF_Chunk_t](#) *chunk, void *opaque)
Function for use with [CF_ChunkList_ComputeGaps\(\)](#)

Functions

- static [CF_ChunkOffset_t](#) [CF_Chunk_MAX](#) ([CF_ChunkOffset_t](#) a, [CF_ChunkOffset_t](#) b)
Selects the larger of the two passed-in offsets.
- void [CF_ChunkListInit](#) ([CF_ChunkList_t](#) *chunks, [CF_ChunkIdx_t](#) max_chunks, [CF_Chunk_t](#) *chunks_mem)
Initialize a CF_ChunkList_t structure.
- void [CF_ChunkListAdd](#) ([CF_ChunkList_t](#) *chunks, [CF_ChunkOffset_t](#) offset, [CF_ChunkSize_t](#) size)

- Public function to add a chunk.*
- void `CF_ChunkListReset` (`CF_ChunkList_t` *chunks)
Resets a chunks structure.
 - void `CF_ChunkList_RemoveFromFirst` (`CF_ChunkList_t` *chunks, `CF_ChunkSize_t` size)
Public function to remove some amount of size from the first chunk.
 - const `CF_Chunk_t` * `CF_ChunkList_GetFirstChunk` (const `CF_ChunkList_t` *chunks)
Public function to get the entire first chunk from the list.
 - uint32 `CF_ChunkList_ComputeGaps` (const `CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` max_gaps, `CF_ChunkSize_t` total, `CF_ChunkOffset_t` start, `CF_ChunkList_ComputeGapFn_t` compute_gap_fn, void *opaque)
Compute gaps between chunks, and call a callback for each.
 - void `CF_Chunks_EraseRange` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` start, `CF_ChunkIdx_t` end)
Erase a range of chunks.
 - void `CF_Chunks_EraseChunk` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` erase_index)
Erase a single chunk.
 - void `CF_Chunks_InsertChunk` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` index_before, const `CF_Chunk_t` *chunk)
Insert a chunk before index_before.
 - `CF_ChunkIdx_t` `CF_Chunks_FindInsertPosition` (`CF_ChunkList_t` *chunks, const `CF_Chunk_t` *chunk)
Finds where a chunk should be inserted in the chunks.
 - int `CF_Chunks_CombinePrevious` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` i, const `CF_Chunk_t` *chunk)
Possibly combines the given chunk with the previous chunk.
 - `CFE_Status_t` `CF_Chunks_CombineNext` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` i, const `CF_Chunk_t` *chunk)
Possibly combines the given chunk with the next chunk.
 - `CF_ChunkIdx_t` `CF_Chunks_FindSmallestSize` (const `CF_ChunkList_t` *chunks)
Finds the smallest size out of all chunks.
 - void `CF_Chunks_Insert` (`CF_ChunkList_t` *chunks, `CF_ChunkIdx_t` i, const `CF_Chunk_t` *chunk)
Insert a chunk.

12.34.1 Detailed Description

The CF Application chunks (spare gap tracking) header file

12.34.2 Typedef Documentation

12.34.2.1 `CF_Chunk_t` `typedef struct CF_Chunk CF_Chunk_t`

Pairs an offset with a size to identify a specific piece of a file.

12.34.2.2 `CF_ChunkIdx_t` `typedef uint32 CF_ChunkIdx_t`

Definition at line 31 of file cf_chunk.h.

12.34.2.3 `CF_ChunkList_ComputeGapFn_t` `typedef void(* CF_ChunkList_ComputeGapFn_t) (const CF_ChunkList_t` `*cs, const CF_Chunk_t *chunk, void *opaque)`

Function for use with `CF_ChunkList_ComputeGaps()`

Parameters

<i>cs</i>	Pointer to the CF_ChunkList_t object
<i>chunk</i>	Pointer to the chunk being currently processed
<i>opaque</i>	Opaque pointer passed through from initial call

Definition at line 63 of file cf_chunk.h.

12.34.2.4 CF_ChunkList_t `typedef struct CF_ChunkList CF_ChunkList_t`
A list of CF_Chunk_t pairs.

This list is ordered by chunk offset, from lowest to highest

12.34.2.5 CF_ChunkOffset_t `typedef uint32 CF_ChunkOffset_t`
Definition at line 32 of file cf_chunk.h.

12.34.2.6 CF_ChunkSize_t `typedef uint32 CF_ChunkSize_t`

Definition at line 33 of file cf_chunk.h.

12.34.3 Function Documentation

12.34.3.1 CF_Chunk_MAX() `static CF_ChunkOffset_t CF_Chunk_MAX (`
`CF_ChunkOffset_t a,`
`CF_ChunkOffset_t b) [inline], [static]`

Selects the larger of the two passed-in offsets.

Parameters

<i>a</i>	First chunk offset
<i>b</i>	Second chunk offset

Returns

the larger CF_ChunkOffset_t value

Definition at line 72 of file cf_chunk.h.

Referenced by CF_Chunks_CombineNext().

12.34.3.2 CF_ChunkList_ComputeGaps() `uint32 CF_ChunkList_ComputeGaps (`
`const CF_ChunkList_t * chunks,`
`CF_ChunkIdx_t max_gaps,`
`CF_ChunkSize_t total,`
`CF_ChunkOffset_t start,`
`CF_ChunkList_ComputeGapFn_t compute_gap_fn,`
`void * opaque)`

Compute gaps between chunks, and call a callback for each.

Description

This function walks over all chunks and computes the gaps between. It can exit early if the calculated gap start is larger than the desired total.

Assumptions, External Events, and Notes:

chunks must not be NULL. compute_gap_fn is a valid function address.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>max_gaps</i>	Maximum number of gaps to compute
<i>total</i>	Size of the entire file
<i>start</i>	Beginning offset for gap computation
<i>compute_gap_fn</i>	Callback function to be invoked for each gap
<i>opaque</i>	Opaque pointer to pass through to callback function

Returns

The number of computed gaps.

Definition at line 362 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_ChunkList::count, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_CFDP_R2Complete(), and CF_CFDP_R_SubstateSendNak().

12.34.3.3 CF_ChunkList_GetFirstChunk() const CF_Chunk_t* CF_ChunkList_GetFirstChunk (const CF_ChunkList_t * chunks)

Public function to get the entire first chunk from the list.

This returns the first chunk from the list, or NULL if the list is empty.

Note

The chunk remains on the list - this call does not consume or remove the chunk from the list.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Returns

Pointer to first chunk from the CF_ChunkList_t object

Return values

NULL	if the list was empty
------	-----------------------

Definition at line 325 of file cf_chunk.c.

References CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_CFDP_S_CheckAndRespondNak().

```
12.34.3.4 CF_ChunkList_RemoveFromFirst() void CF_ChunkList_RemoveFromFirst (
    CF_ChunkList_t * chunks,
    CF_ChunkSize_t size )
```

Public function to remove some amount of size from the first chunk.

Description

This may remove the chunk entirely. This function is to satisfy the use-case where data is retrieved from the structure in-order and once consumed should be removed.

Assumptions, External Events, and Notes:

chunks must not be NULL and list must not be empty

Note

Good computer science would have a generic remove function, but that's much more complex than we need for the use case. We aren't trying to make chunks a general purpose reusable module, so just take the simple case that we need.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>size</i>	Size to remove

Definition at line 299 of file cf_chunk.c.

References CF_Chunks_EraseChunk(), CF_ChunkList::chunks, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_CFDP_S_CheckAndRespondNak().

Here is the call graph for this function:



```
12.34.3.5 CF_ChunkListAdd() void CF_ChunkListAdd (
    CF_ChunkList_t * chunks,
    CF_ChunkOffset_t offset,
    CF_ChunkSize_t size )
```

Public function to add a chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

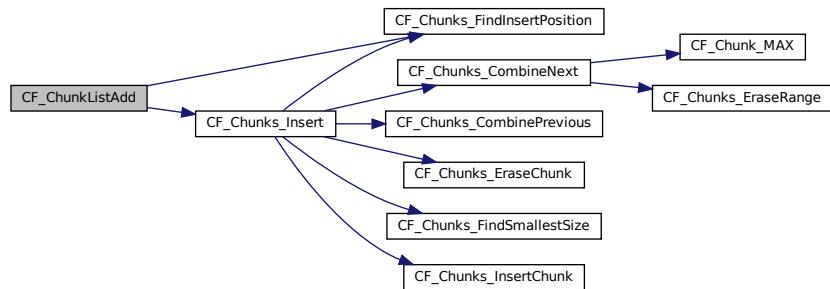
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>offset</i>	Offset of chunk to add
<i>size</i>	Size of chunk to add

Definition at line 280 of file cf_chunk.c.

References CF_ASSERT, CF_Chunks_FindInsertPosition(), and CF_Chunks_Insert().

Referenced by CF_CFDP_R2_SubstateRecvFileData(), and CF_CFDP_S2_Nak().

Here is the call graph for this function:



12.34.3.6 CF_ChunkListInit() void CF_ChunkListInit (

- CF_ChunkList_t * chunks,
- CF_ChunkIdx_t max_chunks,
- CF_Chunk_t * chunks_mem)

Initialize a CF_ChunkList_t structure.

Assumptions, External Events, and Notes:

chunks must not be NULL. chunks_mem must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object to initialize
<i>max_chunks</i>	Maximum number of entries in the chunks_mem array
<i>chunks_mem</i>	Array of CF_Chunk_t objects with length of max_chunks

Definition at line 336 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkListReset(), CF_ChunkList::chunks, and CF_ChunkList::max_chunks.

Referenced by CF_CFDP_InitEngine().

Here is the call graph for this function:



12.34.3.7 CF_ChunkListReset() `void CF_ChunkListReset (`
 `CF_ChunkList_t * chunks)`

Resets a chunks structure.

All chunks are removed from the list, but the max_chunks and chunk memory pointers are retained. This returns the chunk list to the same state as it was after the initial call to [CF_ChunkListInit\(\)](#).

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
---------------------	----------------------------------

Definition at line 350 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_ChunkList::max_chunks.

Referenced by CF_ChunkListInit().

12.34.3.8 CF_Chunks_CombineNext() `CFE_Status_t CF_Chunks_CombineNext (`
 `CF_ChunkList_t * chunks,`
 `CF_ChunkIdx_t i,`
 `const CF_Chunk_t * chunk)`

Possibly combines the given chunk with the next chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>i</code>	Index of chunk to combine
<code>chunk</code>	Chunk data to combine

Returns

boolean code indicating if chunks were combined

Return values

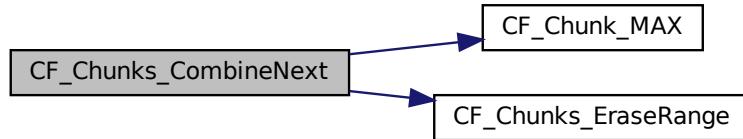
<code>1</code>	if combined with another chunk
<code>0</code>	if not combined

Definition at line 170 of file cf_chunk.c.

References CF_ASSERT, CF_Chunk_MAX(), CF_Chunks_EraseRange(), CF_ChunkList::chunks, CF_ChunkList::count, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

Here is the call graph for this function:



12.34.3.9 CF_Chunks_CombinePrevious()

```
int CF_Chunks_CombinePrevious (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t i,
    const CF_Chunk_t * chunk )
```

Possibly combines the given chunk with the previous chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Index of chunk to combine
<i>chunk</i>	Chunk data to combine

Returns

boolean code indicating if chunks were combined

Return values

1	if combined with another chunk
0	if not combined

Definition at line 133 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_Chunk::offset, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

12.34.3.10 CF_Chunks_EraseChunk()

```
void CF_Chunks_EraseChunk (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t erase_index )
```

Erase a single chunk.

Note

This changes the chunk IDs of all chunks that follow items in the list after the `erase_index` will be shifted/moved to close the gap.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>erase_index</i>	chunk ID to erase

Definition at line 63 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_ChunkList_RemoveFromFirst(), and CF_Chunks_Insert().

```
12.34.3.11 CF_Chunks_EraseRange() void CF_Chunks_EraseRange (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t start,
    CF_ChunkIdx_t end )
```

Erase a range of chunks.

Items in the list starting at the end index will be shifted/moved to close the gap. If end <= start nothing will be done (OK to pass matching start/end as a no-op) If end == list size, list from start on will be erased (nothing moved)

Example: list = {a, b, c, d, e} EraseRange index start 1, end 3 list = {a, d, e}

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>start</i>	Starting chunk ID to erase (inclusive)
<i>end</i>	Ending chunk ID (exclusive, this chunk will not be erased)

Definition at line 45 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, and CF_ChunkList::count.

Referenced by CF_Chunks_CombineNext().

```
12.34.3.12 CF_Chunks_FindInsertPosition() CF_ChunkIdx_t CF_Chunks_FindInsertPosition (
    CF_ChunkList_t * chunks,
    const CF_Chunk_t * chunk )
```

Finds where a chunk should be inserted in the chunks.

Description

This is a C version of std::lower_bound from C++ algorithms.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>chunk</i>	Chunk data to insert

Returns

an index to the first chunk that is greater than or equal to the requested's offset.

Definition at line 101 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_Chunk::offset.

Referenced by CF_ChunkListAdd(), and CF_Chunks_Insert().

12.34.3.13 CF_Chunks_FindSmallestSize() `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (`
`const CF_ChunkList_t * chunks)`

Finds the smallest size out of all chunks.

Assumptions, External Events, and Notes:

chunks must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
---------------	----------------------------------

Returns

The chunk index with the smallest size.

Return values

<i>0</i>	if the chunk list is empty
----------	----------------------------

Definition at line 214 of file cf_chunk.c.

References CF_ChunkList::chunks, CF_ChunkList::count, and CF_Chunk::size.

Referenced by CF_Chunks_Insert().

12.34.3.14 CF_Chunks_Insert() `void CF_Chunks_Insert (`
`CF_ChunkList_t * chunks,`
`CF_ChunkIdx_t i,`
`const CF_Chunk_t * chunk)`

Insert a chunk.

Description

Inserts the chunk at the specified location. May combine with an existing chunk if contiguous.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

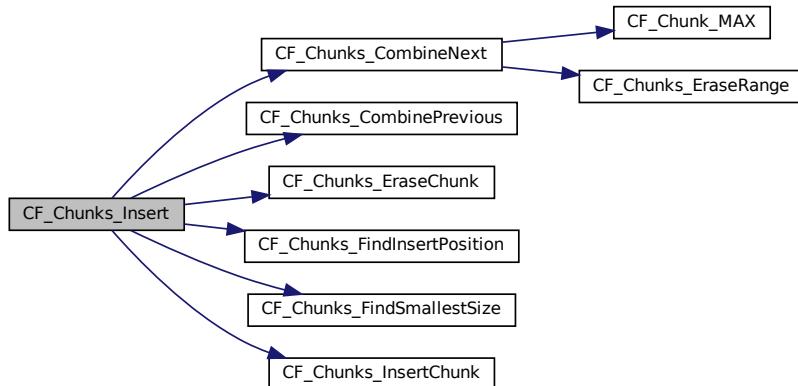
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Position to insert chunk at
<i>chunk</i>	Chunk data to insert

Definition at line 236 of file cf_chunk.c.

References CF_Chunks_CombineNext(), CF_Chunks_CombinePrevious(), CF_Chunks_EraseChunk(), CF_Chunks→_FindInsertPosition(), CF_Chunks_FindSmallestSize(), CF_Chunks_InsertChunk(), CF_ChunkList::chunks, CF→_ChunkList::count, CF_ChunkList::max_chunks, and CF_Chunk::size.

Referenced by CF_ChunkListAdd().

Here is the call graph for this function:

**12.34.3.15 CF_Chunks_InsertChunk()**

```
void CF_Chunks_InsertChunk (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t index_before,
    const CF_Chunk_t * chunk )
```

Insert a chunk before index_before.

Note

This changes the chunk IDs of all chunks that follow items in the list after the index_before will be shifted/moved to open a gap.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>index_before</i>	position to insert at - this becomes the ID of the inserted chunk
<i>chunk</i>	Chunk data to insert (copied)

Definition at line 80 of file cf_chunk.c.

References CF_ASSERT, CF_ChunkList::chunks, CF_ChunkList::count, and CF_ChunkList::max_chunks.
Referenced by CF_Chunks_Insert().

12.35 apps/cf/fsw/src/cf_clist.c File Reference

```
#include "cf_clist.h"
#include "cf_assert.h"
```

Functions

- void **CF_CList_InitNode** (**CF_CListNode_t** *node)
Initialize a clist node.
- void **CF_CList_InsertFront** (**CF_CListNode_t** **head, **CF_CListNode_t** *node)
Insert the given node into the front of a list.
- void **CF_CList_InsertBack** (**CF_CListNode_t** **head, **CF_CListNode_t** *node)
Insert the given node into the back of a list.
- **CF_CListNode_t** * **CF_CList_Pop** (**CF_CListNode_t** **head)
Remove the first node from a list and return it.
- void **CF_CList_Remove** (**CF_CListNode_t** **head, **CF_CListNode_t** *node)
Remove the given node from the list.
- void **CF_CList_InsertAfter** (**CF_CListNode_t** **head, **CF_CListNode_t** *start, **CF_CListNode_t** *after)
Insert the given node into the last after the given start node.
- void **CF_CList_Traverse** (**CF_CListNode_t** *start, **CF_CListFn_t** fn, void *context)
Traverse the entire list, calling the given function on all nodes.
- void **CF_CList_Traverse_R** (**CF_CListNode_t** *end, **CF_CListFn_t** fn, void *context)
Reverse list traversal, starting from end, calling given function on all nodes.

12.35.1 Detailed Description

The CF Application circular list definition source file

This is a circular doubly-linked list implementation. It is used for all data structures in CF.

This file is intended to be a generic class that can be used in other apps.

12.35.2 Function Documentation

12.35.2.1 **CF_CList_InitNode()** void **CF_CList_InitNode** (**CF_CListNode_t** * node)

Initialize a clist node.

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<i>node</i>	Pointer to node structure to be initialized
-------------	---

Definition at line 40 of file cf_clist.c.

References CF_CListNode::next, and CF_CListNode::prev.

Referenced by CF_CFDP_InitEngine(), CF_CList_Remove(), and CF_FreeTransaction().

```
12.35.2.2 CF_CList_InsertAfter() void CF_CList_InsertAfter (
    CF_CListNode_t ** head,
    CF_CListNode_t * start,
    CF_CListNode_t * after )
```

Insert the given node into the last after the given start node.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to remove from
<i>start</i>	Pointer to node to insert
<i>after</i>	Pointer to position to insert after

Definition at line 167 of file cf_clist.c.

References CF_Assert, CF_CListNode::next, and CF_CListNode::prev.

Referenced by CF_CList_InsertAfter_Ex().

```
12.35.2.3 CF_CList_InsertBack() void CF_CList_InsertBack (
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the back of a list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 81 of file cf_clist.c.

References CF_Assert, CF_CListNode::next, and CF_CListNode::prev.

Referenced by CF_CFDP_InitEngine(), CF_CFDP_ResetTransaction(), CF_CList_InsertBack_Ex(), and CF_MoveTransaction().

```
12.35.2.4 CF_CList_InsertFront() void CF_CList_InsertFront (
    CF_CListNode_t ** head,
```

```
CF_CListNode_t * node )
```

Insert the given node into the front of a list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 52 of file cf_clist.c.

References CF_ASSERT, CF_CListNode::next, and CF_CListNode::prev.

12.35.2.5 CF_CList_Pop() `CF_CListNode_t* CF_CList_Pop (`

```
CF_CListNode_t ** head )
```

Remove the first node from a list and return it.

Assumptions, External Events, and Notes:

head must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to remove from
-------------	--

Returns

The first node (now removed) in the list

Return values

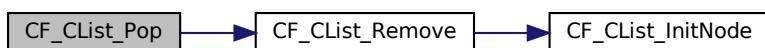
<i>NULL</i>	if list was empty.
-------------	--------------------

Definition at line 111 of file cf_clist.c.

References CF_ASSERT, and CF_CList_Remove().

Referenced by CF_CFDP_FindUnusedChunks().

Here is the call graph for this function:



12.35.2.6 CF_CList_Remove() `void CF_CList_Remove (`

```
CF_CListNode_t ** head,
CF_CListNode_t * node )
```

Remove the given node from the list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to remove from
<i>node</i>	Pointer to node to remove

Definition at line 132 of file cf_clist.c.

References CF_ASSERT, CF_CList_InitNode(), CF_CListNode::next, and CF_CListNode::prev.

Referenced by CF_CList_Pop(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), and CF_MoveTransaction().

Here is the call graph for this function:



12.35.2.7 CF_CList_Traverse() void CF_CList_Traverse (

```
    CF_CListNode_t * start,
    CF_CListFn_t fn,
    void * context )
```

Traverse the entire list, calling the given function on all nodes.

Assumptions, External Events, and Notes:

start may be NULL, fn must be a valid function, context may be NULL.

Note

on traversal it's ok to delete the current node, but do not delete other nodes in the same list!!

Parameters

<i>start</i>	List to traverse (first node)
<i>fn</i>	Callback function to invoke for each node
<i>context</i>	Opaque pointer to pass to callback

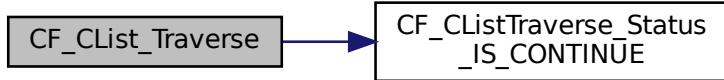
Definition at line 188 of file cf_clist.c.

References CF_CListTraverse_Status_IS_CONTINUE(), and CF_CListNode::next.

Referenced by CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_TickTransactions(), CF_DoPurge←

Queue(), CF_FindTransactionBySequenceNumber(), CF_TraverseAllTransactions(), CF_WriteHistoryQueueDataToFile(), and CF_WriteTxnQueueDataToFile().

Here is the call graph for this function:



12.35.2.8 CF_CList_Traverse_R() `void CF_CList_Traverse_R (`
 `CF_CListNode_t * end,`
 `CF_CListFn_t fn,`
 `void * context)`

Reverse list traversal, starting from end, calling given function on all nodes.

Assumptions, External Events, and Notes:

end may be NULL. fn must be a valid function, context may be NULL.

Note

traverse_R will work backwards from the parameter's prev, and end on param

Parameters

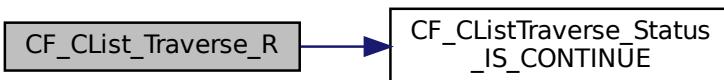
<code>end</code>	List to traverse (last node)
<code>fn</code>	Callback function to invoke for each node
<code>context</code>	Opaque pointer to pass to callback

Definition at line 227 of file cf_clist.c.

References CF_CListTraverse_Status_IS_CONTINUE(), and CF_CListNode::prev.

Referenced by CF_InsertSortPrio().

Here is the call graph for this function:



12.36 apps/cf/fsw/src/cf_clist.h File Reference

```
#include <stddef.h>
#include <stdbool.h>
```

Data Structures

- struct [CF_CListNode](#)

Node link structure.

Macros

- #define [CF_CLIST_CONT](#) [CF_CListTraverse_Status_CONTINUE](#)
Constant indicating to continue traversal.
- #define [CF_CLIST_EXIT](#) [CF_CListTraverse_Status_EXIT](#)
Constant indicating to stop traversal.
- #define [container_of](#)(ptr, type, member) ((type *)((char *)(ptr) - (char *)offsetof(type, member)))
Obtains a pointer to the parent structure.

Typedefs

- typedef struct [CF_CListNode](#) [CF_CListNode_t](#)
Circular linked list node links.
- typedef [CF_CListTraverse_Status_t](#) (* [CF_CListFn_t](#)) ([CF_CListNode_t](#) *node, void *context)
Callback function type for use with [CF_CList_Traverse\(\)](#)

Enumerations

- enum [CF_CListTraverse_Status_t](#) { [CF_CListTraverse_Status_CONTINUE](#) = 0, [CF_CListTraverse_Status_EXIT](#) = 1 }

Functions

- static bool [CF_CListTraverse_Status_IS_CONTINUE](#) ([CF_CListTraverse_Status_t](#) stat)
- void [CF_CList_InitNode](#) ([CF_CListNode_t](#) *node)
Initialize a clist node.
- void [CF_CList_InsertFront](#) ([CF_CListNode_t](#) **head, [CF_CListNode_t](#) *node)
Insert the given node into the front of a list.
- void [CF_CList_InsertBack](#) ([CF_CListNode_t](#) **head, [CF_CListNode_t](#) *node)
Insert the given node into the back of a list.
- void [CF_CList_Remove](#) ([CF_CListNode_t](#) **head, [CF_CListNode_t](#) *node)
Remove the given node from the list.
- [CF_CListNode_t](#) * [CF_CList_Pop](#) ([CF_CListNode_t](#) **head)
Remove the first node from a list and return it.
- void [CF_CList_InsertAfter](#) ([CF_CListNode_t](#) **head, [CF_CListNode_t](#) *start, [CF_CListNode_t](#) *after)
Insert the given node into the last after the given start node.
- void [CF_CList_Traverse](#) ([CF_CListNode_t](#) *start, [CF_CListFn_t](#) fn, void *context)
Traverse the entire list, calling the given function on all nodes.
- void [CF_CList_Traverse_R](#) ([CF_CListNode_t](#) *end, [CF_CListFn_t](#) fn, void *context)
Reverse list traversal, starting from end, calling given function on all nodes.

12.36.1 Detailed Description

The CF Application circular list header file

12.36.2 Macro Definition Documentation

12.36.2.1 CF_CLIST_CONT `#define CF_CLIST_CONT CF_CListTraverse_Status_CONTINUE`

Constant indicating to continue traversal.

Definition at line 38 of file cf_clist.h.

12.36.2.2 CF_CLIST_EXIT `#define CF_CLIST_EXIT CF_CListTraverse_Status_EXIT`

Constant indicating to stop traversal.

Definition at line 39 of file cf_clist.h.

12.36.2.3 container_of `#define container_of(`

```
    ptr,
    type,
    member ) ((type *) ((char *) (ptr) - (char *) offsetof(type, member)))
```

Obtains a pointer to the parent structure.

Given a pointer to a CF_CListNode_t object which is known to be a member of a larger container, this converts the pointer to that of the parent.

Definition at line 69 of file cf_clist.h.

12.36.3 Typedef Documentation

12.36.3.1 CF_CListFn_t `typedef CF_CListTraverse_Status_t (* CF_CListFn_t) (CF_CListNode_t *node,` `void *context)`

Callback function type for use with [CF_CList_Traverse\(\)](#)

Parameters

<code>node</code>	Current node being traversed
<code>context</code>	Opaque pointer passed through from initial call

Returns

integer status code indicating whether to continue traversal

Return values

CF_CLIST_CONT	Indicates to continue traversing the list
CF_CLIST_EXIT	Indicates to stop traversing the list

Definition at line 81 of file cf_clist.h.

12.36.3.2 CF_CListNode_t `typedef struct CF_CListNode CF_CListNode_t`
Circular linked list node links.
Definition at line 61 of file cf_clist.h.

12.36.4 Enumeration Type Documentation

12.36.4.1 CF_CListTraverse_Status_t enum CF_CListTraverse_Status_t

Enumerator

<code>CF_CListTraverse_Status_CONTINUE</code>	
<code>CF_CListTraverse_Status_EXIT</code>	

Definition at line 32 of file cf_clist.h.

12.36.5 Function Documentation

12.36.5.1 CF_CList_InitNode() void CF_CList_InitNode (

`CF_CListNode_t * node`)

Initialize a clist node.

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<code>node</code>	Pointer to node structure to be initialized
-------------------	---

Definition at line 40 of file cf_clist.c.

References `CF_CListNode::next`, and `CF_CListNode::prev`.

Referenced by `CF_CFDP_InitEngine()`, `CF_CList_Remove()`, and `CF_FreeTransaction()`.

12.36.5.2 CF_CList_InsertAfter() void CF_CList_InsertAfter (

`CF_CListNode_t ** head,`
`CF_CListNode_t * start,`
`CF_CListNode_t * after`)

Insert the given node into the last after the given start node.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<code>head</code>	Pointer to head of list to remove from
<code>start</code>	Pointer to node to insert
<code>after</code>	Pointer to position to insert after

Definition at line 167 of file cf_clist.c.
References CF_Assert, CF_CListNode::next, and CF_CListNode::prev.
Referenced by CF_CList_InsertAfter_Ex().

12.36.5.3 CF_CList_InsertBack() void CF_CList_InsertBack (

```
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the back of a list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 81 of file cf_clist.c.
References CF_Assert, CF_CListNode::next, and CF_CListNode::prev.
Referenced by CF_CFDP_InitEngine(), CF_CFDP_ResetTransaction(), CF_CList_InsertBack_Ex(), and CF_Move← Transaction().

12.36.5.4 CF_CList_InsertFront() void CF_CList_InsertFront (

```
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the front of a list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 52 of file cf_clist.c.
References CF_Assert, CF_CListNode::next, and CF_CListNode::prev.

12.36.5.5 CF_CList_Pop() CF_CListNode_t* CF_CList_Pop (

```
    CF_CListNode_t ** head )
```

Remove the first node from a list and return it.

Assumptions, External Events, and Notes:

head must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to remove from
-------------	--

Returns

The first node (now removed) in the list

Return values

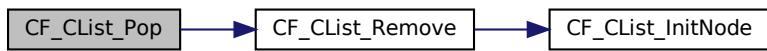
<i>NULL</i>	if list was empty.
-------------	--------------------

Definition at line 111 of file cf_clist.c.

References CF_ASSERT, and CF_CList_Remove().

Referenced by CF_CFDP_FindUnusedChunks().

Here is the call graph for this function:

**12.36.5.6 CF_CList_Remove()** void CF_CList_Remove (

```

    CF_CListNode_t ** head,
    CF_CListNode_t * node
)
```

Remove the given node from the list.

Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

Parameters

<i>head</i>	Pointer to head of list to remove from
<i>node</i>	Pointer to node to remove

Definition at line 132 of file cf_clist.c.

References CF_ASSERT, CF_CList_InitNode(), CF_CListNode::next, and CF_CListNode::prev.

Referenced by CF_CList_Pop(), CF_CList_Remove_Ex(), CF_DequeueTransaction(), and CF_MoveTransaction().

Here is the call graph for this function:



12.36.5.7 CF_CList_Traverse() `void CF_CList_Traverse (`
 `CF_CListNode_t * start,`
 `CF_CListFn_t fn,`
 `void * context)`

Traverse the entire list, calling the given function on all nodes.

Assumptions, External Events, and Notes:

start may be NULL, fn must be a valid function, context may be NULL.

Note

on traversal it's ok to delete the current node, but do not delete other nodes in the same list!!

Parameters

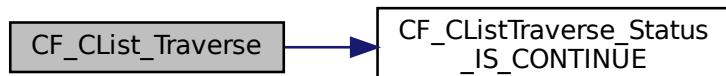
<code>start</code>	List to traverse (first node)
<code>fn</code>	Callback function to invoke for each node
<code>context</code>	Opaque pointer to pass to callback

Definition at line 188 of file cf_clist.c.

References CF_CListTraverse_Status_IS_CONTINUE(), and CF_CListNode::next.

Referenced by CF_CFDP_CycleTx(), CF_CFDP_DisableEngine(), CF_CFDP_TickTransactions(), CF_DoPurgeQueue(), CF_FindTransactionBySequenceNumber(), CF_TraverseAllTransactions(), CF_WriteHistoryQueueDataToFile(), and CF_WriteTxnQueueDataToFile().

Here is the call graph for this function:



```
12.36.5.8 CF_CList_Traverse_R() void CF_CList_Traverse_R (
    CF_CListNode_t * end,
    CF_CListFn_t fn,
    void * context )
```

Reverse list traversal, starting from end, calling given function on all nodes.

Assumptions, External Events, and Notes:

end may be NULL. fn must be a valid function, context may be NULL.

Note

traverse_R will work backwards from the parameter's prev, and end on param

Parameters

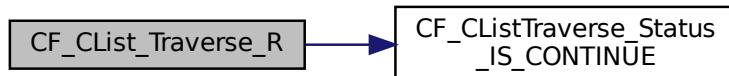
<i>end</i>	List to traverse (last node)
<i>fn</i>	Callback function to invoke for each node
<i>context</i>	Opaque pointer to pass to callback

Definition at line 227 of file cf_clist.c.

References CF_CListTraverse_Status_IS_CONTINUE(), and CF_CListNode::prev.

Referenced by CF_InsertSortPrio().

Here is the call graph for this function:



```
12.36.5.9 CF_CListTraverse_Status_IS_CONTINUE() static bool CF_CListTraverse_Status_IS_CONTINUE (
    CF_CListTraverse_Status_t stat ) [inline], [static]
```

Checks if the list traversal should continue

Definition at line 44 of file cf_clist.h.

References CF_CListTraverse_Status_CONTINUE.

Referenced by CF_CList_Traverse(), and CF_CList_Traverse_R().

12.37 apps/cf/fsw/src(cf_cmd.c File Reference

```
#include "cf_app.h"
#include "cf_verify.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_utils.h"
#include "cf_version.h"
```

```
#include "cf_platform_cfg.h"
#include "cf_cfdp.h"
#include "cf_cmd.h"
#include <string.h>
```

Functions

- **CFE_Status_t CF_NoopCmd** (const **CF_NoopCmd_t** *msg)
The no-operation command.
- **CFE_Status_t CF_ResetCountersCmd** (const **CF_ResetCountersCmd_t** *msg)
The reset counters command.
- **CFE_Status_t CF_TxFileCmd** (const **CF_TxFileCmd_t** *msg)
Ground command to start a file transfer.
- **CFE_Status_t CF_PlaybackDirCmd** (const **CF_PlaybackDirCmd_t** *msg)
Ground command to start directory playback.
- **CF_ChAction_Status_t CF_DoChanAction** (const **CF_UnionArgs_Payload_t** *data, const char *errstr, **CF_ChActionFn_t** fn, void *context)
Common logic for all channel-based commands.
- **CF_ChAction_Status_t CF_DoFreezeThaw** (uint8 chan_num, void *arg)
Channel action to set the frozen bit for a channel.
- **CFE_Status_t CF_FreezeCmd** (const **CF_FreezeCmd_t** *msg)
Freeze a channel.
- **CFE_Status_t CF_ThawCmd** (const **CF_ThawCmd_t** *msg)
Thaw a channel.
- **CF_Transaction_t * CF_FindTransactionBySequenceNumberAllChannels** (**CF_TransactionSeq_t** ts, **CF_EntityId_t** eid)
Search for a transaction across all channels.
- **int32 CF_TsnChanAction** (const **CF_Transaction_Payload_t** *data, const char *cmdstr, **CF_TsnChanAction_fn_t** fn, void *context)
Common logic for all transaction sequence number and channel commands.
- **void CF_DoSuspRes_Txn** (**CF_Transaction_t** *txn, **CF_ChAction_SuspResArg_t** *context)
Set the suspended bit in a transaction.
- **void CF_DoSuspRes** (const **CF_Transaction_Payload_t** *payload, uint8 action)
Handle transaction suspend and resume commands.
- **CFE_Status_t CF_SuspendCmd** (const **CF_SuspendCmd_t** *msg)
Handle transaction suspend command.
- **CFE_Status_t CF_ResumeCmd** (const **CF_ResumeCmd_t** *msg)
Handle transaction resume command.
- **void CF_Cancel_TxnCmd** (**CF_Transaction_t** *txn, void *ignored)
tsn chan action to cancel a transaction.
- **CFE_Status_t CF_CancelCmd** (const **CF_CancelCmd_t** *msg)
Handle a cancel ground command.
- **void CF_Abandon_TxnCmd** (**CF_Transaction_t** *txn, void *ignored)
tsn chan action to abandon a transaction.
- **CFE_Status_t CF_AbandonCmd** (const **CF_AbandonCmd_t** *msg)
Handle an abandon ground command.
- **CF_ChAction_Status_t CF_DoEnableDisableDequeue** (uint8 chan_num, void *arg)

- Sets the dequeue enable/disable flag for a channel.
 - CFE_Status_t CF_EnableDequeueCmd (const CF_EnableDequeueCmd_t *msg)
Handle an enable dequeue ground command.
 - CFE_Status_t CF_DisableDequeueCmd (const CF_DisableDequeueCmd_t *msg)
Handle a disable dequeue ground command.
- CF_ChAction_Status_t CF_DoEnableDisablePolldir (uint8 chan_num, void *arg)
Sets the enable/disable flag for the specified polling directory.
- CFE_Status_t CF_EnableDirPollingCmd (const CF_EnableDirPollingCmd_t *msg)
Enable a polling dir ground command.
- CFE_Status_t CF_DisableDirPollingCmd (const CF_DisableDirPollingCmd_t *msg)
Disable a polling dir ground command.
- CF_CListTraverse_Status_t CF_PurgeHistory (CF_CListNode_t *node, void *arg)
Purge the history queue for the given channel.
- CF_CListTraverse_Status_t CF_PurgeTransaction (CF_CListNode_t *node, void *ignored)
Purge the pending transaction queue.
- CF_ChAction_Status_t CF_DoPurgeQueue (uint8 chan_num, void *arg)
Channel action command to perform purge queue operations.
- CFE_Status_t CF_PurgeQueueCmd (const CF_PurgeQueueCmd_t *msg)
Ground command to purge either the history or pending queues.
- CFE_Status_t CF_WriteQueueCmd (const CF_WriteQueueCmd_t *msg)
Ground command to write a file with queue information.
- CF_ChAction_Status_t CF_ValidateChunkSizeCmd (CF_ChunkSize_t val, uint8 chan_num)
Checks if the value is less than or equal to the max PDU size.
- CF_ChAction_Status_t CF_ValidateMaxOutgoingCmd (uint32 val, uint8 chan_num)
Checks if the value is within allowable range as outgoing packets per wakeup.
- void CF_GetSetParamCmd (bool is_set, CF_GetSet_ValueID_t param_id, uint32 value, uint8 chan_num)
Perform a configuration get/set operation.
- CFE_Status_t CF_SetParamCmd (const CF_SetParamCmd_t *msg)
Ground command to set a configuration parameter.
- CFE_Status_t CF_GetParamCmd (const CF_GetParamCmd_t *msg)
Ground command to set a configuration parameter.
- CFE_Status_t CF_EnableEngineCmd (const CF_EnableEngineCmd_t *msg)
Ground command enable engine.
- CFE_Status_t CF_DisableEngineCmd (const CF_DisableEngineCmd_t *msg)
Ground command disable engine.
- CFE_Status_t CF_SendHkCmd (const CF_SendHkCmd_t *msg)
Send CF housekeeping packet.
- CFE_Status_t CF_WakeupCmd (const CF_WakeupCmd_t *msg)
CF wakeup function.

12.37.1 Detailed Description

The CF Application command handling source file

All ground commands are processed in this file. All supporting functions necessary to process the commands are also here.

12.37.2 Function Documentation

```
12.37.2.1 CF_Abandon_TxnCmd() void CF_Abandon_TxnCmd (
    CF_Transaction_t * txn,
    void * ignored )
```

tsn chan action to abandon a transaction.

This helper function is used with [CF_TsnChanAction\(\)](#) to abandon matched transactions

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

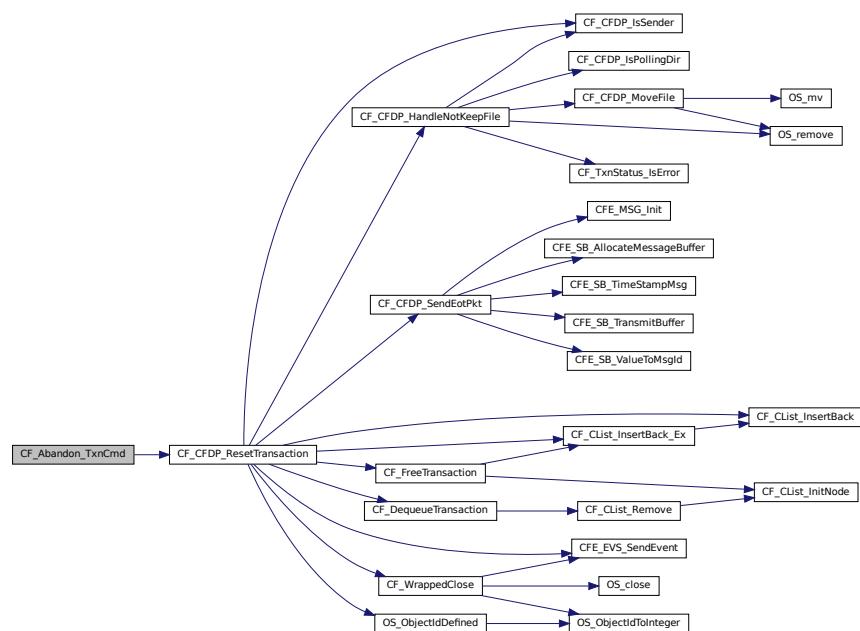
<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 526 of file cf_cmd.c.

References CF_CFDP_ResetTransaction().

Referenced by CF_AbandonCmd().

Here is the call graph for this function:



```
12.37.2.2 CF_AbandonCmd() CFE_Status_t CF_AbandonCmd (
    const CF_AbandonCmd_t * msg )
```

Handle an abandon ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

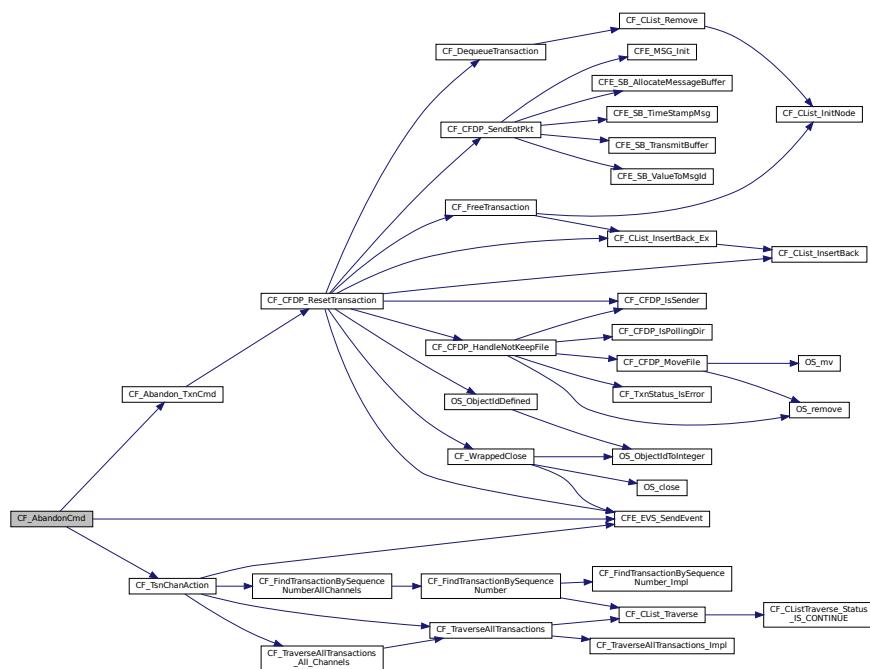
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 537 of file cf_cmd.c.

References CF_Abandon_TxnCmd(), CF_AppData, CF_CMD_ABANDON_CHAN_ERR_EID, CF_CMD_ABANDON_INF_EID, CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CF_E_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_AbandonCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.3 CF_Cancel_TxnCmd() `void CF_Cancel_TxnCmd (CF_Transaction_t * txn,
void * ignored)`

tsn chan action to cancel a transaction.

This helper function is used with [CF_TsnChanAction\(\)](#) to cancel matched transactions

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

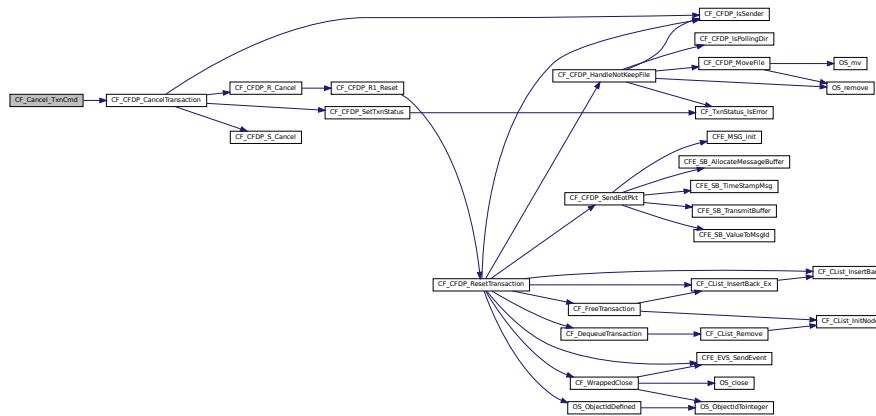
<code>txn</code>	Pointer to transaction object
<code>ignored</code>	Not used by this function

Definition at line 491 of file cf_cmd.c.

References CF_CFDP_CancelTransaction().

Referenced by CF_CancelCmd().

Here is the call graph for this function:



12.37.2.4 CF_CancelCmd() [CFE_Status_t](#) CF_CancelCmd (
 const [CF_CancelCmd_t](#) * msg)

Handle a cancel ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

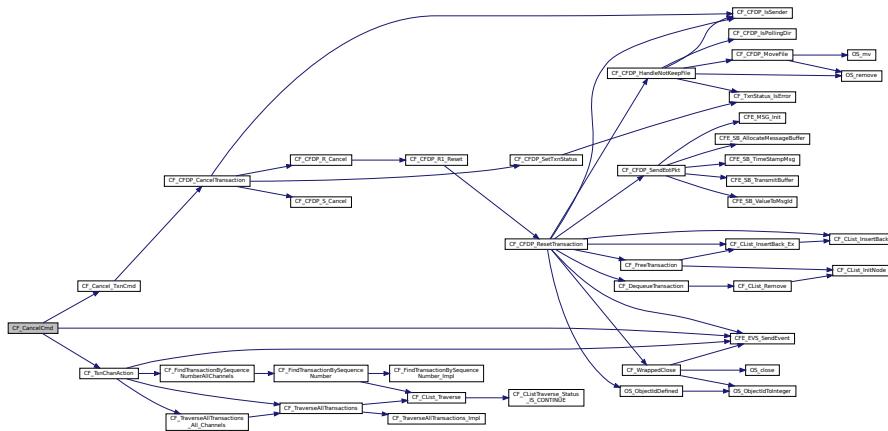
Parameters

msg	Pointer to command message
-----	----------------------------

Definition at line 502 of file cf_cmd.c.

References CF_AppData, CF_Cancel_TxnCmd(), CF_CMD_CANCEL_CHAN_ERR_EID, CF_CMD_CANCEL_IN←F_EID, CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS←SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters←::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_CancelCmd::Payload.

Here is the call graph for this function:



12.37.2.5 CF_DisableDequeueCmd() `CFE_Status_t CF_DisableDequeueCmd (const CF_DisableDequeueCmd_t * msg)`

Handle a disable dequeue ground command.

Assumptions, External Events, and Notes.

msg must not be NULL.

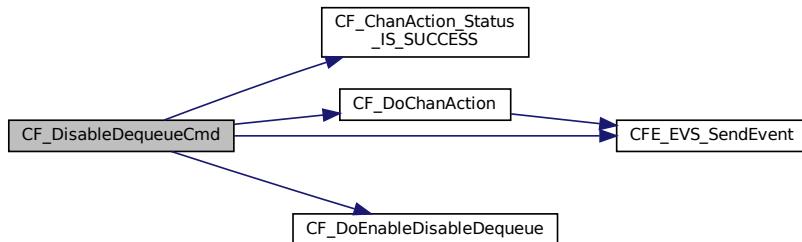
Parameters

msg Pointer to command message

Definition at line 600 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_DISABLE_DEQUEUE_ERR_EID, CF_CMD_DISABLE_DEQUEUE_INF_EID, CF_DoChanAction(), CF_DoEnableDisableDequeue(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_DisableDequeueCmd::Payload.

Here is the call graph for this function:



12.37.2.6 CF_DisableDirPollingCmd() `CFE_Status_t CF_DisableDirPollingCmd (const CF_DisableDirPollingCmd_t * msg)`

Disable a polling dir ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

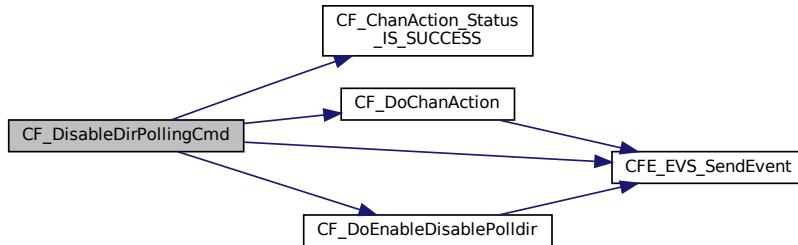
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 685 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_DISABLE_POLLDIR_ERR_EID, CF_CMD_DISABLE_POLLDIR_INF_EID, CF_DoChanAction(), CF_DoEnableDisablePolldir(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_DisableDirPollingCmd::Payload.

Here is the call graph for this function:



12.37.2.7 CF_DisableEngineCmd() `CFE_Status_t CF_DisableEngineCmd (const CF_DisableEngineCmd_t * msg)`

Ground command disable engine.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

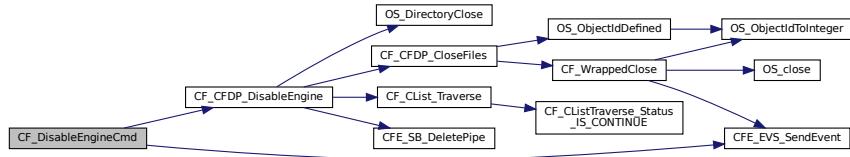
Definition at line 1201 of file cf_cmd.c.

References CF_AppData, CF_CFDP_DisableEngine(), CF_CMD_DISABLE_ENGINE_INF_EID, CF_CMD_ENG_ALREADY_DIS_INF_EID, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Hk

CmdCounters::cmd, CF_HkPacket_Payload::counters, CF_Engine::enabled, CF_AppData_t::engine, CF_AppData_t::hk, and CF_HkPacket::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.8 CF_DoChanAction() CF_ChanAction_Status_t CF_DoChanAction (

```

    const CF_UnionArgs_Payload_t * data,
    const char * errstr,
    CF_ChanActionFn_t fn,
    void * context )
  
```

Common logic for all channel-based commands.

Description

All the commands that act on channels or have the special "all channels" parameter come through this function. This puts all common logic in one place. It does not handle the command accept or reject counters.

Assumptions, External Events, and Notes:

cmd must not be NULL, errstr must not be NULL, fn must be a valid function, context may be NULL.

Parameters

<i>data</i>	Pointer to payload being processed
<i>errstr</i>	String to be included in the EVS event if command should fail
<i>fn</i>	Callback action function to invoke for each affected channel
<i>context</i>	Opaque pointer to pass through to callback (not used in this function)

Returns

The return value from the given action function.

Return values

<i>CF_ChanAction_Status_ERROR</i>	on error
-----------------------------------	----------

Definition at line 233 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_ALL_CHANNELS, CF_ChanAction_Status_ERROR, CF_ChanAction_Status_SUCCESS, CF_CMD_CHAN_PARAM_ERR_EID, CF_NUM_CHANNELS, CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_FreezeCmd(), CF_PurgeQueueCmd(), and CF_ThawCmd().

Here is the call graph for this function:



12.37.2.9 CF_DoEnableDisableDequeue() `CF_ChанAction_Status_t CF_DoEnableDisableDequeue (uint8 chan_num, void * arg)`

Sets the dequeue enable/disable flag for a channel.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be <code>CF_ChанAction_BoolArg_t*</code>

Returns

Always succeeds, so returns `CF_ChанAction_Status_SUCCESS`.

Definition at line 561 of file `cf_cmd.c`.

References `CF_ChанAction_BoolArg::barg`, `CF_AppData`, `CF_ChанAction_Status_SUCCESS`, `CF_ConfigTable::chan`, `CF_AppData_t::config_table`, and `CF_ChannelConfig::dequeue_enabled`.

Referenced by `CF_DisableDequeueCmd()`, and `CF_EnableDequeueCmd()`.

12.37.2.10 CF_DoEnableDisablePolldir() `CF_ChанAction_Status_t CF_DoEnableDisablePolldir (uint8 chan_num, void * arg)`

Sets the enable/disable flag for the specified polling directory.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be <code>CF_ChанAction_BoolMsgArg_t*</code>

Returns

success/fail status code

Return values

<i>CF_ChanAction_Status_SUCCESS</i>	if successful
<i>CF_ChanAction_Status_ERROR</i>	if failed

Definition at line 625 of file cf_cmd.c.

References *CF_ChanAction_BoolMsgArg::barg*, *CF_UnionArgs_Payload::byte*, *CF_ALL_POLLDIRS*, *CF_AppData*, *CF_ChanAction_Status_ERROR*, *CF_ChanAction_Status_SUCCESS*, *CF_CMD_POLLDIR_INVALID_ERR_EID*, *C_F_MAX_POLLING_DIR_PER_CHAN*, *CFE_EVS_EventType_ERROR*, *CFE_EVS_SendEvent()*, *CF_ConfigTable::chan*, *CF_AppData_t::config_table*, *CF_ChanAction_BoolMsgArg::data*, *CF_PollDir::enabled*, and *CF_ChannelConfig::polldir*.

Referenced by *CF_DisableDirPollingCmd()*, and *CF_EnableDirPollingCmd()*.

Here is the call graph for this function:

**12.37.2.11 CF_DoFreezeThaw()** *CF_ChanAction_Status_t* *CF_DoFreezeThaw* (

```

    uint8 chan_num,
    void * arg
)
```

Channel action to set the frozen bit for a channel.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be <i>CF_ChanAction_BoolArg_t*</i>

Returns

Always succeeds, so returns *CF_ChanAction_Status_SUCCESS*.

Definition at line 269 of file cf_cmd.c.

References *CF_ChanAction_BoolArg::barg*, *CF_AppData*, *CF_ChanAction_Status_SUCCESS*, *CF_HkPacket::Payload::channel_hk*, *CF_HkChannel_Data::frozen*, *CF_AppData_t::hk*, and *CF_HkPacket::Payload*.

Referenced by *CF_FreezeCmd()*, and *CF_ThawCmd()*.

```
12.37.2.12 CF_DoPurgeQueue() CF_ChанAction_Status_t CF_DoPurgeQueue (
    uint8 chan_num,
    void * arg )
```

Channel action command to perform purge queue operations.

Description

Determines from the command parameters which queues to traverse and purge state.

Assumptions, External Events, and Notes:

None

Parameters

<i>chan_num</i>	CF channel number
<i>arg</i>	Pointer to purge queue command

Returns

integer status code indicating success or failure

Return values

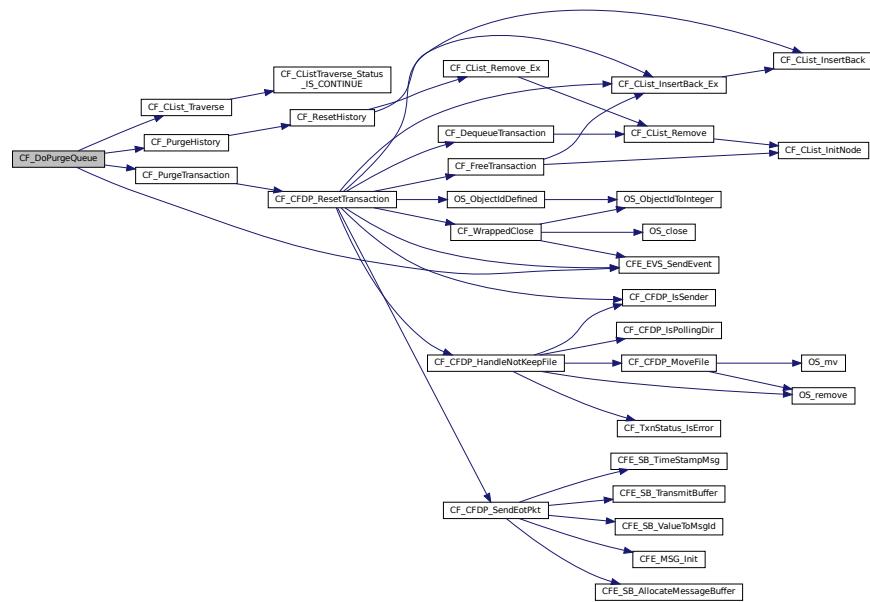
<i>CF_ChанAction_Status_SUCCESS</i>	if successful
<i>CF_ChанAction_Status_ERROR</i>	on error

Definition at line 739 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_AppData, CF_ChанAction_Status_ERROR, CF_ChанAction_Status_SUCCESS, CF_CList_Traverse(), CF_CMD_PURGE_ARG_ERR_EID, CF_PurgeHistory(), CF_PurgeTransaction(), CF_QueueIdx_HIST, CF_QueueIdx_PEND, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Engine::channels, CF_AppData_t::engine, and CF_Channel::qs.

Referenced by CF_PurgeQueueCmd().

Here is the call graph for this function:



```

12.37.2.13 CF_DoSuspRes() void CF_DoSuspRes (
    const CFE_Transaction_Payload_t * payload,
    uint8 action )
  
```

Handle transaction suspend and resume commands.

Description

This is called for both suspend and resume ground commands. It uses the [CF_TsnChanAction\(\)](#) function to perform the command.

Assumptions, External Events, and Notes:

payload must not be NULL.

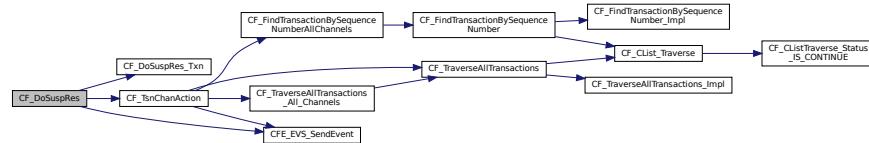
Parameters

<i>payload</i>	Pointer to the command message
<i>action</i>	Action to take (suspend or resume)

Definition at line 426 of file cf_cmd.c.

References CF_AppData, CF_CMD_SUSPRES_CHAN_ERR_EID, CF_CMD_SUSPRES_INF_EID, CF_CMD_SUSPRES_SAME_INF_EID, CF_DoSuspRes_Txn(), CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVT_INFORMATION, CFE_EVS_SendEvent(), CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_ChанAction_SuspResArg::same. Referenced by CF_ResumeCmd(), and CF_SuspendCmd().

Here is the call graph for this function:



12.37.2.14 CF_DoSuspRes_Txn() `void CF_DoSuspRes_Txn (`
`CF_Transaction_t * txn,`
`CF_ChanAction_SuspResArg_t * context)`

Set the suspended bit in a transaction.

Assumptions, External Events, and Notes:

txn must not be NULL. context must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>context</i>	Pointer to CF_ChanAction_SuspResArg_t structure from initial call

Definition at line 407 of file cf_cmd.c.

References CF_ChanAction_SuspResArg::action, CF_Assert, CF_StateFlags::com, CF_Transaction::flags, CF_ChanAction_SuspResArg::same, and CF_Flags_Common::suspended.

Referenced by CF_DoSuspRes().

12.37.2.15 CF_EnableDequeueCmd() `CFE_Status_t CF_EnableDequeueCmd (`
`const CF_EnableDequeueCmd_t * msg)`

Handle an enable dequeue ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

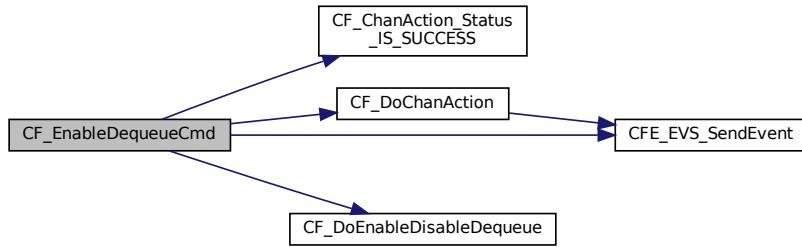
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 575 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_ENABLE_DEQUEUE_ERR_EID, CF_CMD_ENABLE_DEQUEUE_INF_EID, CF_DoChanAction(), CF_DoEnableDisableDequeue(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_EnableDequeueCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.16 CF_EnableDirPollingCmd() `CFE_Status_t CF_EnableDirPollingCmd (`
`const CF_EnableDirPollingCmd_t * msg)`

Enable a polling dir ground command.

Assumptions, External Events, and Notes:

`msg` must not be NULL.

Parameters

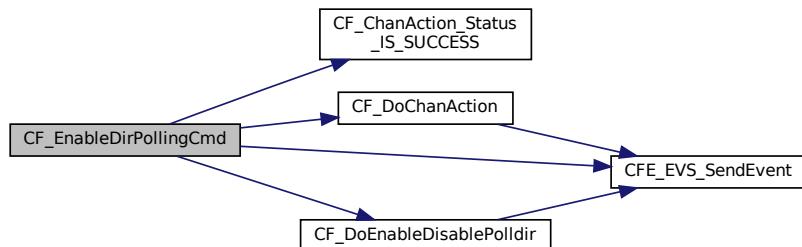
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 658 of file cf_cmd.c.

References `CF_AppData`, `CF_ChanAction_Status_IS_SUCCESS()`, `CF_CMD_ENABLE_POLLDIR_ERR_EID`, `CF_CMD_ENABLE_POLLDIR_INF_EID`, `CF_DoChanAction()`, `CF_DoEnableDisablePolldir()`, `CFE_EVS_EventType_EROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_HkCmdCounters::cmd`, `CF_HkPacket_Payload::counters`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_EnableDirPollingCmd::Payload`.

Referenced by `CF_ProcessGroundCommand()`.

Here is the call graph for this function:



12.37.2.17 CF_EnableEngineCmd() `CFE_Status_t CF_EnableEngineCmd (const CF_EnableEngineCmd_t * msg)`

Ground command enable engine.

Assumptions, External Events, and Notes:

msg must not be NULL.

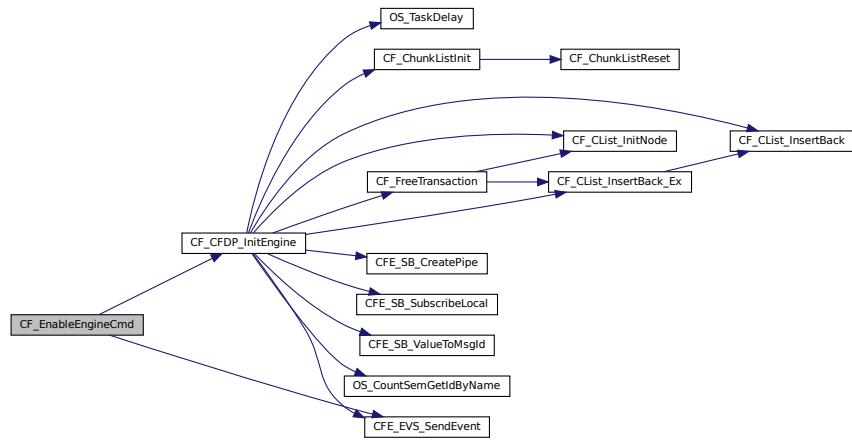
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1169 of file cf_cmd.c.

References CF_AppData, CF_CFDP_InitEngine(), CF_CMD_ENABLE_ENGINE_ERR_EID, CF_CMD_ENABLE_ENGINE_INF_EID, CF_CMD_ENG_ALREADY_ENA_INF_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_Engine::enabled, CF_AppData_t::engine, CF_HkCmdCounters::err, CF_AppData_t::hk, and CF_HkPacket::Payload.

Here is the call graph for this function:



12.37.2.18 CF_FindTransactionBySequenceNumberAllChannels() `CF_Transaction_t* CF_FindTransactionBySequenceNumberAllChannels (CF_TransactionSeq_t ts, CF_EntityId_t eid)`

Search for a transaction across all channels.

Assumptions, External Events, and Notes:

None

Parameters

<code>ts</code>	Transaction sequence number to find
<code>eid</code>	Entity ID of the transaction

Returns

Pointer to the transaction, if found

Return values

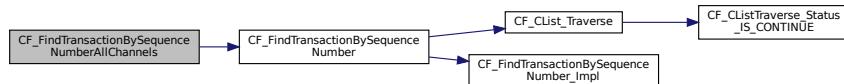
<code>NULL</code>	if transaction not found
-------------------	--------------------------

Definition at line 331 of file cf_cmd.c.

References CF_AppData, CF_FindTransactionBySequenceNumber(), CF_NUM_CHANNELS, CF_Engine::channels, and CF_AppData_t::engine.

Referenced by CF_TsnChanAction().

Here is the call graph for this function:



12.37.2.19 CF_FreezeCmd() `CFE_Status_t CF_FreezeCmd(const CF_FreezeCmd_t * msg)`

Freeze a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

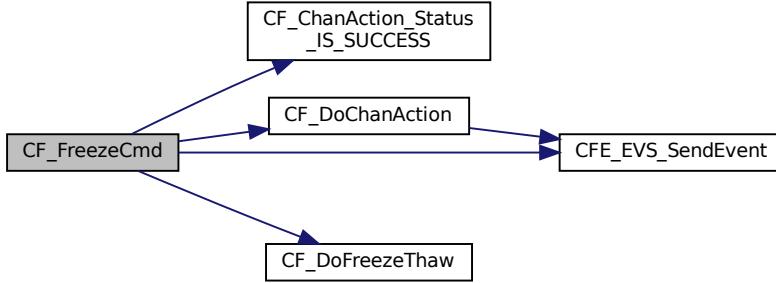
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 283 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_FREEZE_ERR_EID, CF_CMD_FREEZE_INF_EID, CF_DoChanAction(), CF_DoFreezeThaw(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_FreezeCmd::Payload.

Here is the call graph for this function:



12.37.2.20 CF_GetParamCmd() `CFE_Status_t CF_GetParamCmd (`
`const CF_GetParamCmd_t * msg)`

Ground command to set a configuration parameter.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

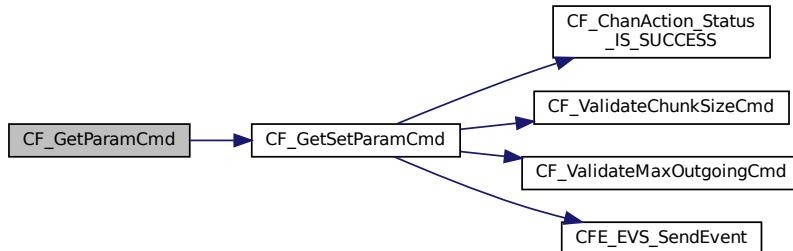
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1154 of file cf_cmd.c.

References CF_SetParamCmd(), CFE_SUCCESS, CF_GetParam_Payload::chan_num, CF_GetParam_Payload::key, and CF_GetParamCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



```
12.37.2.21 CF_GetSetParamCmd() void CF_GetSetParamCmd (
    bool is_set,
    CF_GetSet_ValueID_t param_id,
    uint32 value,
    uint8 chan_num )
```

Perform a configuration get/set operation.

For a set, this sets the value within the CF configuration. For a get, this generates an EVS event with the requested information.

Description

Combine get and set in one function with common logic.

Assumptions, External Events, and Notes:

None

Parameters

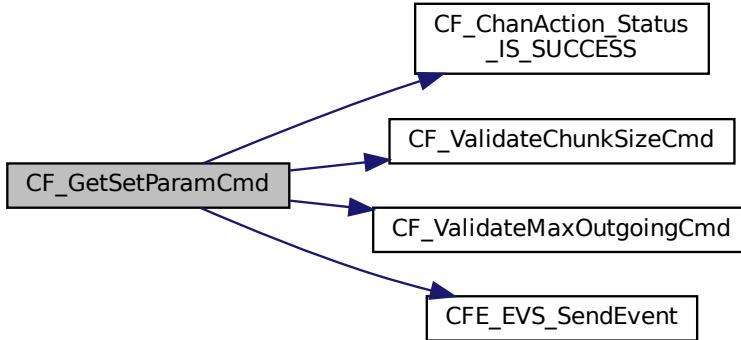
<i>is_set</i>	Whether to get (false) or set (true)
<i>param_id</i>	Parameter ID
<i>value</i>	Value to get/set
<i>chan_num</i>	Channel number to operate on

Definition at line 984 of file cf_cmd.c.

References CF_ChannelConfig::ack_limit, CF_ChannelConfig::ack_timer_s, CF_AppData, CF_ChAction_Status_IS_SUCCESS(), CF_CMD_GETSET1_INF_EID, CF_CMD_GETSET2_INF_EID, CF_CMD_GETSET_CHAN_ERR_EID, CF_CMD_GETSET_PARAM_ERR_EID, CF_CMD_GETSET_VALIDATE_ERR_EID, CF_ERROR, CF_GetSet_ValueID_ack_limit, CF_GetSet_ValueID_ack_timer_s, CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup, CF_GetSet_ValueID_inactivity_timer_s, CF_GetSet_ValueID_local_eid, CF_GetSet_ValueID_nak_limit, CF_GetSet_ValueID_nak_timer_s, CF_GetSet_ValueID_outgoing_file_chunk_size, CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup, CF_GetSet_ValueID_ticks_per_second, CF_NUM_CHANNELS, CF_ValidateChunkSizeCmd(), CF_ValidateMaxOutgoingCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_ConfigTable::chan, CF_HkCmdCounters::cmd, CF_AppData_t::config_table, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_ChannelConfig::inactivity_timer_s, CF_ConfigTable::local_eid, CF_ChannelConfig::max_outgoing_messages_per_wakeup, CF_ChannelConfig::nak_limit, CF_ChannelConfig::nak_timer_s, CF_ConfigTable::outgoing_file_chunk_size, CF_HkPacket::Payload, CF_ConfigTable::rx_crc_calc_bytes_per_wakeup, and CF_ConfigTable::ticks_per_second.

Referenced by CF_GetParamCmd(), and CF_SetParamCmd().

Here is the call graph for this function:



12.37.2.22 CF_NoopCmd() `CFE_Status_t CF_NoopCmd (`
`const CF_NoopCmd_t * msg)`

The no-operation command.

Description

This function has a signature the same of all cmd_ functions. This function simply prints an event message. Increments the command accept counter. The msg parameter is ignored in this one.

Assumptions, External Events, and Notes:

None

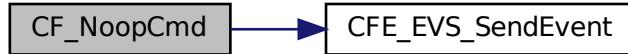
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 48 of file cf_cmd.c.

References CF_AppData, CF_MAJOR_VERSION, CF_MINOR_VERSION, CF_MISSION_REV, CF_NOOP_INF_E_ID, CF_REVISION, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_AppData_t::hk, and CF_HkPacket::Payload.

Here is the call graph for this function:



12.37.2.23 CF_PlaybackDirCmd() `CF_Status_t CF_PlaybackDirCmd (const CF_PlaybackDirCmd_t * msg)`

Ground command to start directory playback.

Description

This function has a signature the same of all cmd_ functions. Increments the command accept or reject counter.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

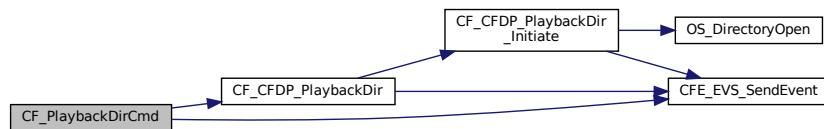
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 183 of file cf_cmd.c.

References CF_AppData, CF_CFDP_CLASS_1, CF_CFDP_CLASS_2, CF_CFDP_PlaybackDir(), CF_CMD_BAD_← PARAM_ERR_EID, CF_CMD_PLAYBACK_DIR_ERR_EID, CF_CMD_PLAYBACK_DIR_INF_EID, CF_NUM_CHAN← NELS, CF_TxFile_Payload::cfdp_class, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CF← E_EVS_SendEvent(), CFE_SUCCESS, CF_TxFile_Payload::chan_num, CF_HkCmdCounters::cmd, CF_HkPacket← _Payload::counters, CF_TxFile_Payload::dest_id, CF_TxFile_Payload::dst_filename, CF_HkCmdCounters::err, C← F_AppData_t::hk, CF_TxFile_Payload::keep, CF_HkPacket::Payload, CF_PlaybackDirCmd::Payload, CF_TxFile← Payload::priority, and CF_TxFile_Payload::src_filename.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.24 CF_PurgeHistory() `CF_CListTraverse_Status_t CF_PurgeHistory (`

```
CF_CListNode_t * node,
void * arg )
```

Purge the history queue for the given channel.

This helper function is used in conjunction with [CF_CList_Traverse\(\)](#)

Assumptions, External Events, and Notes:

node must not be NULL. chan must not be NULL.

Parameters

<i>node</i>	Current list node being traversed
<i>arg</i>	Channel pointer passed through as opaque object, must be CF_Channel_t*

Returns

Always [CF_CLIST_CONT](#) to process all entries

Definition at line 712 of file cf_cmd.c.

References [CF_CLIST_CONT](#), [CF_ResetHistory\(\)](#), and [container_of](#).

Referenced by [CF_DoPurgeQueue\(\)](#).

Here is the call graph for this function:



12.37.2.25 CF_PurgeQueueCmd() [CFE_Status_t CF_PurgeQueueCmd \(const CF_PurgeQueueCmd_t * msg \)](#)

Ground command to purge either the history or pending queues.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

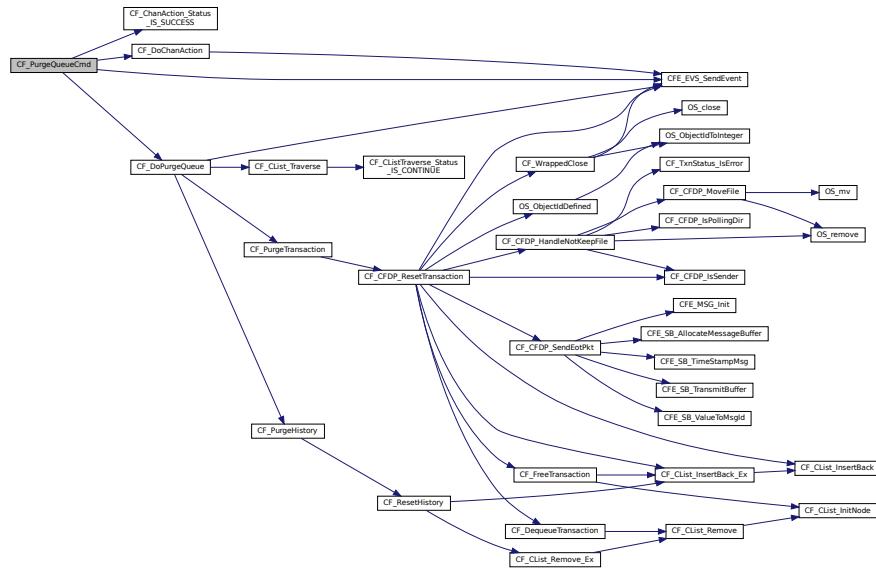
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 790 of file cf_cmd.c.

References [CF_AppData](#), [CF_ChAction_Status_IS_SUCCESS\(\)](#), [CF_CMD_PURGE_QUEUE_ERR_EID](#), [CF_CMD_PURGE_QUEUE_INF_EID](#), [CF_DoChanAction\(\)](#), [CF_DoPurgeQueue\(\)](#), [CFE_EVT_EventType_ERROR](#), [CFE_EVT_EventType_INFORMATION](#), [CFE_EVT_SendEvent\(\)](#), [CFE_SUCCESS](#), [CF_HkCmdCounters::cmd](#), [CF_HkPacket_Payload::counters](#), [CF_HkCmdCounters::err](#), [CF_AppData_t::hk](#), [CF_HkPacket::Payload](#), and [CF_PurgeQueueCmd::Payload](#).

Referenced by [CF_ProcessGroundCommand\(\)](#).

Here is the call graph for this function:



12.37.2.26 CF_PurgeTransaction() *CF_CListTraverse_Status_t* CF_PurgeTransaction (

```
CF_CListNode_t * node,  
void * ignored )
```

Purge the pending transaction queue.

This helper function is used in conjunction with [CF_CList_Traverse\(\)](#)

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<i>node</i>	Current list node being traversed
<i>ignored</i>	Not used by this implementation

Returns

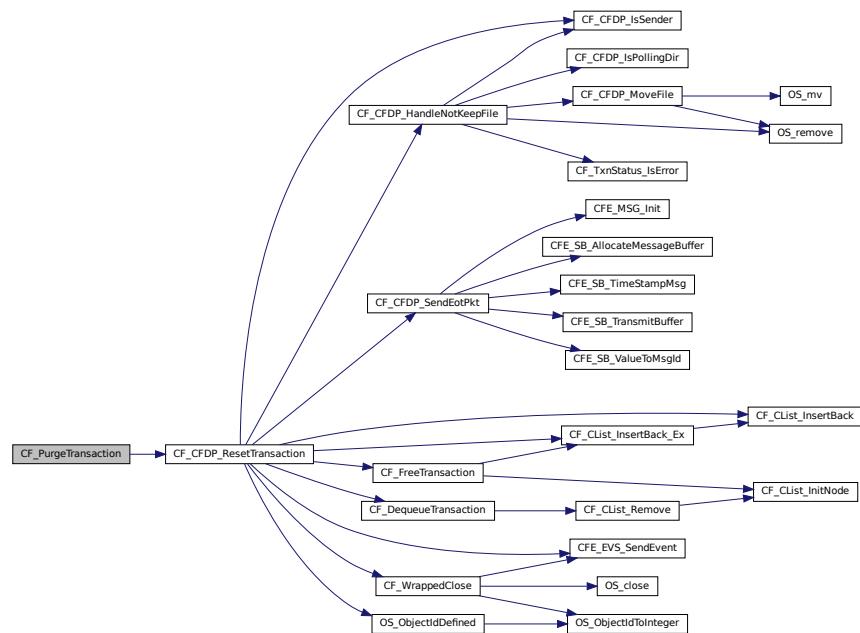
Always [CF_CLIST_CONT](#) to process all entries

Definition at line 726 of file cf_cmd.c.

References CF_CFDP_ResetTransaction(), CF_CLIST_CONT, and container_of.

Referenced by CF_DoPurgeQueue().

Here is the call graph for this function:



12.37.2.27 CF_ResetCountersCmd() [CFE_Status_t](#) CF_ResetCountersCmd (
 const [CF_ResetCountersCmd_t](#) * msg)

The reset counters command.

Description

This function has a signature the same of all cmd_ functions. Resets the given counter, or all. Increments the command accept or reject counter. If the command counters are reset, then there is no increment.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 64 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_AppData, CF_CMD_RESET_INVALID_ERR_EID, CF_NUM_CHAN< NELS, CF_Reset_all, CF_Reset_command, CF_Reset_down, CF_Reset_fault, CF_RESET_INF_EID, CF_Reset<_

up, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkPacket_Payload::channel_hk, CF_HkCmdCounters::cmd, CF_HkChannel_Data::counters, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_HkCounters::fault, CF_AppData_t::hk, CF_HkPacket::Payload, CF_ResetCountersCmd::Payload, CF_HkCounters::recv, and CF_HkCounters::sent.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.28 CF_ResumeCmd() `CFE_Status_t CF_ResumeCmd (`
`const CF_ResumeCmd_t * msg)`

Handle transaction resume command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

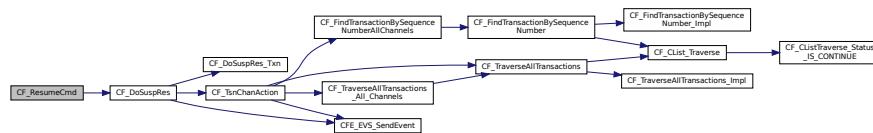
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 479 of file cf_cmd.c.

References CF_DoSuspRes(), CFE_SUCCESS, and CF_ResumeCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.29 CF_SendHkCmd() `CFE_Status_t CF_SendHkCmd (`
`const CF_SendHkCmd_t * msg)`

Send CF housekeeping packet.

Description

The command to send the CF housekeeping packet

Assumptions, External Events, and Notes:

None

Parameters

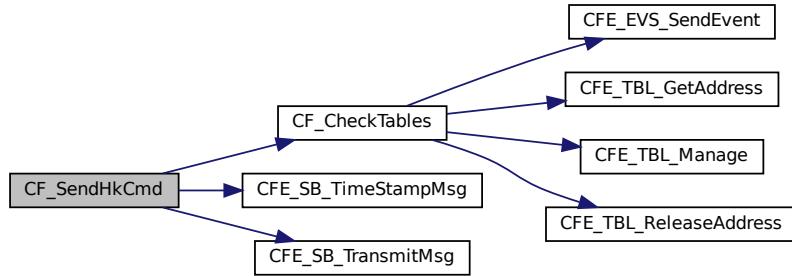
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1225 of file cf_cmd.c.

References CF_AppData, CF_CheckTables(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SUCCESS, CF_AppData_t::hk, and CF_HkPacket::TelemetryHeader.

Referenced by CF_AppPipe().

Here is the call graph for this function:

**12.37.2.30 CF_SetParamCmd()**

```
CFE_Status_t CF_SetParamCmd(
```

```
    const CF_SetParamCmd_t * msg )
```

Ground command to set a configuration parameter.

Assumptions, External Events, and Notes:

`msg` must not be NULL.

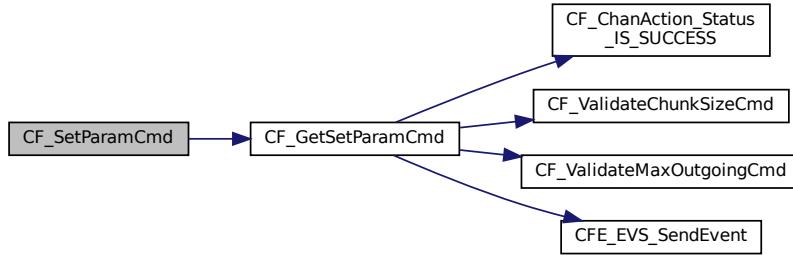
Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1139 of file cf_cmd.c.

References CF_GetSetParamCmd(), CFE_SUCCESS, CF_SetParam_Payload::chan_num, CF_SetParam_Payload::key, CF_SetParamCmd::Payload, and CF_SetParam_Payload::value.

Here is the call graph for this function:



12.37.2.31 CF_SuspendCmd() `CFE_Status_t CF_SuspendCmd (const CF_SuspendCmd_t * msg)`

Handle transaction suspend command.

Assumptions, External Events, and Notes:

msg must not be NULL.

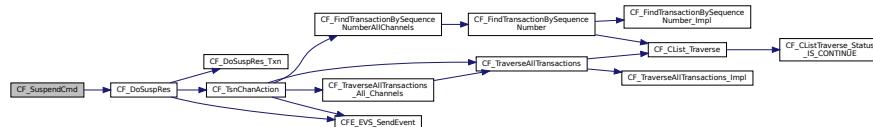
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 467 of file cf_cmd.c.

References CF_DoSuspRes(), CFE_SUCCESS, and CF_SuspendCmd::Payload.

Here is the call graph for this function:



12.37.2.32 CF_ThawCmd() `CFE_Status_t CF_ThawCmd (const CF_ThawCmd_t * msg)`

Thaw a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

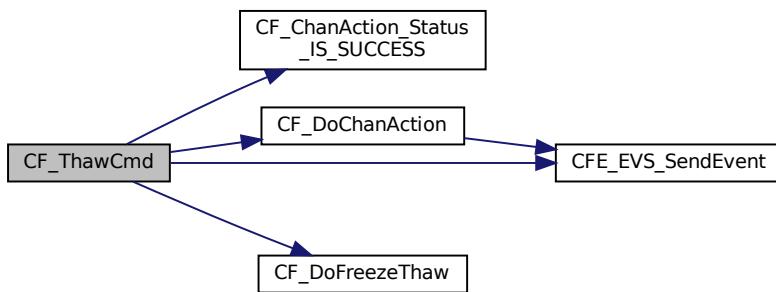
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 307 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_THAW_ERR_EID, CF_CMD_THAW_INF_EID, CF_DoChanAction(), CF_DoFreezeThaw(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_ThawCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.37.2.33 CF_TsnChanAction() `int32 CF_TsnChanAction (`
 `const CF_Transaction_Payload_t * data,`
 `const char * cmdstr,`
 `CF_TsnChanAction_fn_t fn,`
 `void * context)`

Common logic for all transaction sequence number and channel commands.

Description

All the commands that on a transaction on a particular channel come through this function. This puts all common logic in one place. It does handle the command accept or reject counters.

Assumptions, External Events, and Notes:

`cmd` must not be NULL, `fn` must be a valid function, `context` may be NULL.

Parameters

<code>data</code>	Pointer to payload being processed
<code>cmdstr</code>	String to include in any generated EVS events
<code>fn</code>	Callback function to invoke for each matched transaction
<code>context</code>	Opaque object to pass through to the callback

Returns

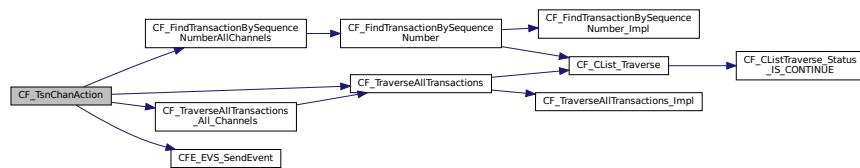
returns the number of transactions acted upon

Definition at line 359 of file cf_cmd.c.

References CF_ALL_CHANNELS, CF_AppData, CF_CMD_TRANS_NOT_FOUND_ERR_EID, CF_CMD_TSN_CH_AN_INVALID_ERR_EID, CF_COMPOUND_KEY, CF_FindTransactionBySequenceNumberAllChannels(), CF_NUM_CHANNELS, CF_TraverseAllTransactions(), CF_TraverseAllTransactions_All_Channels(), CFE_EVS_EventType_ER_ROR, CFE_EVS_SendEvent(), CF_Transaction_Payload::chan, CF_Engine::channels, CF_Transaction_Payload::eid, CF_AppData_t::engine, and CF_Transaction_Payload::ts.

Referenced by CF_AbandonCmd(), CF_CancelCmd(), and CF_DoSuspRes().

Here is the call graph for this function:

**12.37.2.34 CF_TxFileCmd()** `CFE_Status_t CF_TxFileCmd(`

`const CF_TxFileCmd_t * msg)`

Ground command to start a file transfer.

Description

This function has a signature the same of all cmd_ functions. Increments the command accept or reject counter.

Assumptions, External Events, and Notes:

msg must not be NULL.

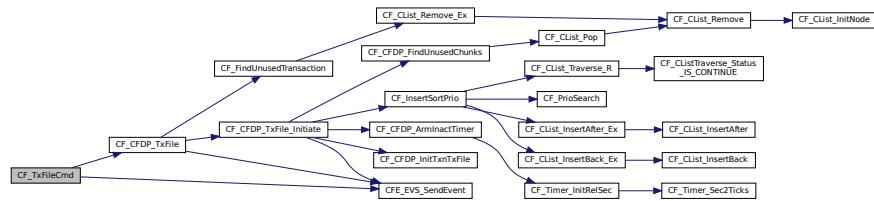
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 134 of file cf_cmd.c.

References CF_AppData, CF_CFDP_CLASS_1, CF_CFDP_CLASS_2, CF_CFDP_TxFile(), CF_CMD_BAD_PARAMETER_EID, CF_CMD_TX_FILE_ERR_EID, CF_CMD_TX_FILE_INF_EID, CF_NUM_CHANNELS, CF_TxFile_Payload::cfdp_class, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_TxFile_Payload::chan_num, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_TxFile_Payload::dest_id, CF_TxFile_Payload::dst_filename, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_TxFile_Payload::keep, CF_HkPacket::Payload, CF_TxFileCmd::Payload, CF_TxFile_Payload::priority, and CF_TxFile_Payload::src_filename.

Here is the call graph for this function:



12.37.2.35 CF_ValidateChunkSizeCmd() `CF_ChanAction_Status_t CF_ValidateChunkSizeCmd (CF_ChunkSize_t val, uint8 chan_num)`

Checks if the value is less than or equal to the max PDU size.

Assumptions, External Events, and Notes:

None

Parameters

<code>val</code>	Size of chunk to test
<code>chan_num</code>	Ignored by this implementation

Returns

status code indicating if check passed

Return values

<code>CF_ChanAction_Status_SUCCESS</code>	if successful (val is less than or equal to max PDU)
<code>CF_ChanAction_Status_ERROR</code>	if failed (val is greater than max PDU)

Definition at line 949 of file cf_cmd.c.

References `CF_ChanAction_Status_ERROR`, and `CF_ChanAction_Status_SUCCESS`.

Referenced by `CF_GetSetParamCmd()`.

12.37.2.36 CF_ValidateMaxOutgoingCmd() `CF_ChanAction_Status_t CF_ValidateMaxOutgoingCmd (uint32 val, uint8 chan_num)`

Checks if the value is within allowable range as outgoing packets per wakeup.

Assumptions, External Events, and Notes:

None

Parameters

<i>val</i>	Number to test
<i>chan_num</i>	CF channel number

Returns

status code indicating if check passed

Return values

<i>CF_ChanAction_Status_SUCCESS</i>	if successful (val is allowable as max packets per wakeup)
<i>CF_ChanAction_Status_ERROR</i>	if failed (val is not allowed)

Definition at line 965 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_ERROR, CF_ChanAction_Status_SUCCESS, CF_ConfigTable<::chan, CF_AppData_t::config_table, and CF_ChannelConfig::sem_name.

Referenced by CF_GetSetParamCmd().

12.37.2.37 CF_WakeupCmd() *CFE_Status_t* *CF_WakeupCmd* (

```
const CF_WakeupCmd_t * msg )
```

CF wakeup function.

Description

Performs a single engine cycle for each wakeup

Assumptions, External Events, and Notes:

None

Parameters

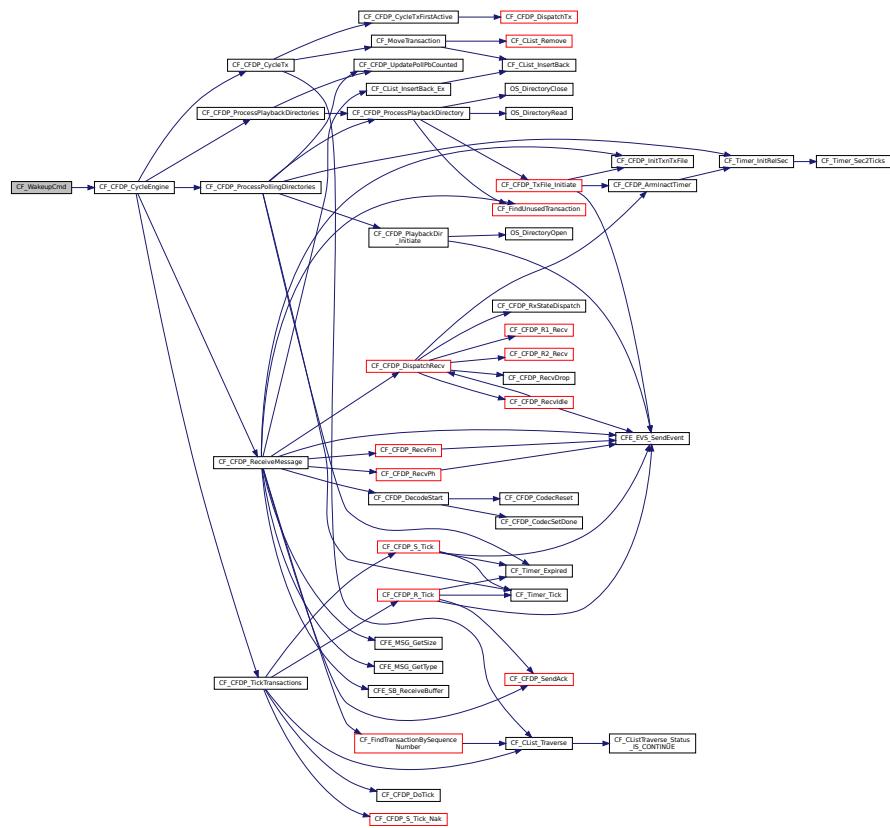
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1242 of file cf_cmd.c.

References CF_CFDP_CycleEngine(), CF_PERF_ID_CYCLE_ENG, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and CFE_SUCCESS.

Referenced by CF_AppPipe().

Here is the call graph for this function:



```
12.37.2.38 CF_WriteQueueCmd() CFE_Status_t CF_WriteQueueCmd ( const CF_WriteQueueCmd_t * msg )
```

Ground command to write a file with queue information.

Assumptions, External Events, and Notes

msg must not be `None`.

Parameters

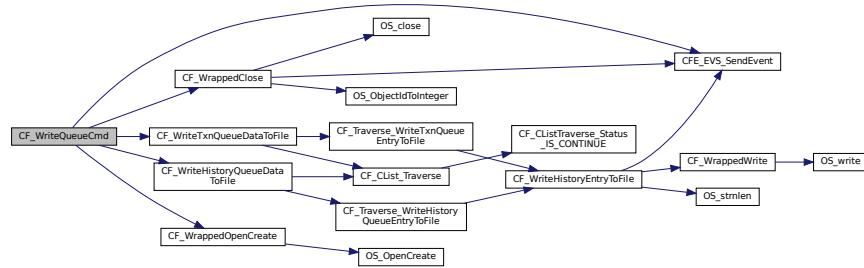
msg Pointer to command message

Definition at line 813 of file cf_cmd.c.

References CF_AppData, CF_CMD_WQ_ARGS_ERR_EID, CF_CMD_WQ_CHAN_ERR_EID, CF_CMD_WQ_INF_EID, CF_CMD_WQ_OPEN_ERR_EID, CF_CMD_WQ_WRITEHIST_RX_ERR_EID, CF_CMD_WQ_WRITEHIST_TX_ERR_EID, CF_CMD_WQ_WRITEQ_PEND_ERR_EID, CF_CMD_WQ_WRITEQ_RX_ERR_EID, CF_CMD_WQ_WRITEQ_TX_ERR_EID, CF_Direction_RX, CF_Direction_TX, CF_NUM_CHANNELS, CF_Queue_active, CF_Queue_all, CF_Queue_history, CF_Queue_pend, CF_QueueIdx_PEND, CF_QueueIdx_RX, CF_QueueIdx_TXA, CF_Queue_Idx_TXW, CF_Type_all, CF_Type_down, CF_Type_up, CF_WrappedClose(), CF_WrappedOpenCreate(), CF_WriteHistoryQueueDataToFile(), CF_WriteTxnQueueDataToFile(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_Error

_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_WriteQueue_Payload::chan, CF_Engine::channels, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_AppData_t::engine, CF_HkCmdCounters::err, CF_WriteQueue_Payload::filename, CF_AppData_t::hk, OS_FILE_FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_OBJECT_ID_UNDEFINED, OS_WRITE_ONLY, CF_HkPacket::Payload, CF_WriteQueueCmd::Payload, CF_WriteQueue_Payload::queue, and CF_WriteQueue_Payload::type.

Here is the call graph for this function:



12.38 apps/cf/fsw/src(cf_cmd.h File Reference

```
#include "cfe.h"
#include "cf_app.h"
#include "cf_utils.h"
```

Data Structures

- struct [CF_ChanAction_BooleanArg](#)
An object to use with channel-scope actions requiring only a boolean argument.
- struct [CF_ChanAction_SuspResArg](#)
An object to use with channel-scope actions for suspend/resume.
- struct [CF_ChanAction_BooleanMsgArg](#)
An object to use with channel-scope actions that require the message value.
- struct [CF_ChanAction_MsgArg](#)
An object to use with channel-scope actions that require the message value.

Typedefs

- typedef [CF_ChanAction_Status_t](#)(* [CF_ChanActionFn_t](#)) (uint8 chan_num, void *context)
A callback function for use with [CF_DoChanAction\(\)](#)
- typedef struct [CF_ChanAction_BooleanArg](#) [CF_ChanAction_BooleanArg_t](#)
An object to use with channel-scope actions requiring only a boolean argument.
- typedef [CF_TraverseAllTransactions_fn_t](#) [CF_TsnChanAction_fn_t](#)
A callback to use with transaction actions.
- typedef struct [CF_ChanAction_SuspResArg](#) [CF_ChanAction_SuspResArg_t](#)
An object to use with channel-scope actions for suspend/resume.
- typedef struct [CF_ChanAction_BooleanMsgArg](#) [CF_ChanAction_BooleanMsgArg_t](#)
An object to use with channel-scope actions that require the message value.
- typedef struct [CF_ChanAction_MsgArg](#) [CF_ChanAction_MsgArg_t](#)
An object to use with channel-scope actions that require the message value.

Enumerations

- enum `CF_ChanAction_Status_t` { `CF_ChanAction_Status_SUCCESS` = 0, `CF_ChanAction_Status_ERROR` = -1 }

Functions

- static bool `CF_ChanAction_Status_IS_SUCCESS` (`CF_ChanAction_Status_t` stat)
• `CFE_Status_t CF_SendHkCmd` (const `CF_SendHkCmd_t` *msg)
Send CF housekeeping packet.
- `CFE_Status_t CF_WakeupCmd` (const `CF_WakeupCmd_t` *msg)
CF wakeup function.
- `CFE_Status_t CF_NoopCmd` (const `CF_NoopCmd_t` *msg)
The no-operation command.
- `CFE_Status_t CF_ResetCountersCmd` (const `CF_ResetCountersCmd_t` *msg)
The reset counters command.
- `CFE_Status_t CF_TxFileCmd` (const `CF_TxFileCmd_t` *msg)
Ground command to start a file transfer.
- `CFE_Status_t CF_PlaybackDirCmd` (const `CF_PlaybackDirCmd_t` *msg)
Ground command to start directory playback.
- `CF_ChanAction_Status_t CF_DoChanAction` (const `CF_UnionArgs_Payload_t` *data, const char *errstr, `CF_ChanActionFn_t` fn, void *context)
Common logic for all channel-based commands.
- `CF_ChanAction_Status_t CF_DoFreezeThaw` (uint8 chan_num, void *arg)
Channel action to set the frozen bit for a channel.
- `CFE_Status_t CF_FreezeCmd` (const `CF_FreezeCmd_t` *msg)
Freeze a channel.
- `CFE_Status_t CF_ThawCmd` (const `CF_ThawCmd_t` *msg)
Thaw a channel.
- `CF_Transaction_t * CF_FindTransactionBySequenceNumberAllChannels` (`CF_TransactionSeq_t` ts, `CF_EntityId_t` eid)
Search for a transaction across all channels.
- `int32 CF_TsnChanAction` (const `CF_Transaction_Payload_t` *data, const char *cmdstr, `CF_TsnChanAction_fn_t` fn, void *context)
Common logic for all transaction sequence number and channel commands.
- void `CF_DoSuspRes_Txn` (`CF_Transaction_t` *txn, `CF_ChanAction_SuspResArg_t` *context)
Set the suspended bit in a transaction.
- void `CF_DoSuspRes` (const `CF_Transaction_Payload_t` *payload, uint8 action)
Handle transaction suspend and resume commands.
- `CFE_Status_t CF_SuspendCmd` (const `CF_SuspendCmd_t` *msg)
Handle transaction suspend command.
- `CFE_Status_t CF_ResumeCmd` (const `CF_ResumeCmd_t` *msg)
Handle transaction resume command.
- void `CF_Cancel_TxnCmd` (`CF_Transaction_t` *txn, void *ignored)
tsn chan action to cancel a transaction.
- `CFE_Status_t CF_CancelCmd` (const `CF_CancelCmd_t` *msg)
Handle a cancel ground command.
- void `CF_Abandon_TxnCmd` (`CF_Transaction_t` *txn, void *ignored)
tsn chan action to abandon a transaction.

- `CFE_Status_t CF_AbandonCmd (const CF_AbandonCmd_t *msg)`
Handle an abandon ground command.
- `CF_ChAction_Status_t CF_DoEnableDisableDequeue (uint8 chan_num, void *arg)`
Sets the dequeue enable/disable flag for a channel.
- `CFE_Status_t CF_EnableDequeueCmd (const CF_EnableDequeueCmd_t *msg)`
Handle an enable dequeue ground command.
- `CFE_Status_t CF_DisableDequeueCmd (const CF_DisableDequeueCmd_t *msg)`
Handle a disable dequeue ground command.
- `CF_ChAction_Status_t CF_DoEnableDisablePolldir (uint8 chan_num, void *arg)`
Sets the enable/disable flag for the specified polling directory.
- `CFE_Status_t CF_EnableDirPollingCmd (const CF_EnableDirPollingCmd_t *msg)`
Enable a polling dir ground command.
- `CFE_Status_t CF_DisableDirPollingCmd (const CF_DisableDirPollingCmd_t *msg)`
Disable a polling dir ground command.
- `CF_CListTraverse_Status_t CF_PurgeHistory (CF_CListNode_t *node, void *arg)`
Purge the history queue for the given channel.
- `CF_CListTraverse_Status_t CF_PurgeTransaction (CF_CListNode_t *node, void *ignored)`
Purge the pending transaction queue.
- `CF_ChAction_Status_t CF_DoPurgeQueue (uint8 chan_num, void *arg)`
Channel action command to perform purge queue operations.
- `CFE_Status_t CF_PurgeQueueCmd (const CF_PurgeQueueCmd_t *msg)`
Ground command to purge either the history or pending queues.
- `CFE_Status_t CF_WriteQueueCmd (const CF_WriteQueueCmd_t *msg)`
Ground command to write a file with queue information.
- `CF_ChAction_Status_t CF_ValidateChunkSizeCmd (CF_ChunkSize_t val, uint8 chan_num)`
Checks if the value is less than or equal to the max PDU size.
- `CF_ChAction_Status_t CF_ValidateMaxOutgoingCmd (uint32 val, uint8 chan_num)`
Checks if the value is within allowable range as outgoing packets per wakeup.
- `void CF_GetSetParamCmd (bool is_set, CF_GetSet_ValueID_t param_id, uint32 value, uint8 chan_num)`
Perform a configuration get/set operation.
- `CFE_Status_t CF_SetParamCmd (const CF_SetParamCmd_t *msg)`
Ground command to set a configuration parameter.
- `CFE_Status_t CF_GetParamCmd (const CF_GetParamCmd_t *msg)`
Ground command to set a configuration parameter.
- `CFE_Status_t CF_EnableEngineCmd (const CF_EnableEngineCmd_t *msg)`
Ground command enable engine.
- `CFE_Status_t CF_DisableEngineCmd (const CF_DisableEngineCmd_t *msg)`
Ground command disable engine.

12.38.1 Detailed Description

CF command processing function declarations

12.38.2 Typedef Documentation

12.38.2.1 CF_ChanAction_BoolArg_t `typedef struct CF_ChanAction_BoolArg CF_ChanAction_BoolArg_t`
An object to use with channel-scope actions requiring only a boolean argument.

12.38.2.2 CF_ChanAction_BoolMsgArg_t `typedef struct CF_ChanAction_BoolMsgArg CF_ChanAction_BoolMsgArg_t`
An object to use with channel-scope actions that require the message value.
This combines a boolean action arg with the command message value

12.38.2.3 CF_ChanAction_MsgArg_t `typedef struct CF_ChanAction_MsgArg CF_ChanAction_MsgArg_t`
An object to use with channel-scope actions that require the message value.
This combines a boolean action arg with the command message value

12.38.2.4 CF_ChanAction_SuspResArg_t `typedef struct CF_ChanAction_SuspResArg CF_ChanAction_SuspResArg_t`
An object to use with channel-scope actions for suspend/resume.
This combines a boolean action arg with an output that indicates if it was a change or not.

12.38.2.5 CF_ChanActionFn_t `typedef CF_ChanAction_Status_t (* CF_ChanActionFn_t) (uint8 chan_num,
void *context)`
A callback function for use with `CF_DoChanAction()`

Parameters

<code>chan_num</code>	The CF channel number, for statistics purposes
<code>context</code>	Opaque object passed through from initial call

Definition at line 53 of file cf_cmd.h.

12.38.2.6 CF_TsnChanAction_fn_t `typedef CF_TraverseAllTransactions_fn_t CF_TsnChanAction_fn_t`
A callback to use with transaction actions.

For now this is the same as `CF_TraverseAllTransactions_fn_t`

Definition at line 68 of file cf_cmd.h.

12.38.3 Enumeration Type Documentation

12.38.3.1 CF_ChanAction_Status_t `enum CF_ChanAction_Status_t`

Enumerator

<code>CF_ChanAction_Status_SUCCESS</code>	
<code>CF_ChanAction_Status_ERROR</code>	

Definition at line 33 of file cf_cmd.h.

12.38.4 Function Documentation

12.38.4.1 CF_Abandon_TxnCmd() `void CF_Abandon_TxnCmd (`
`CF_Transaction_t * txn,`

```
void * ignored )
```

tsn chan action to abandon a transaction.

This helper function is used with [CF_TsnChanAction\(\)](#) to abandon matched transactions

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

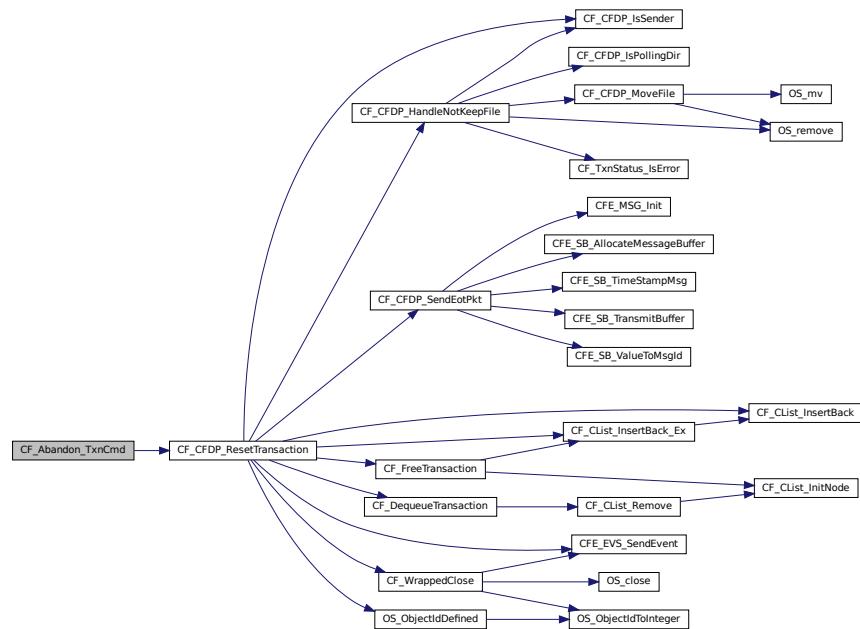
<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 526 of file cf_cmd.c.

References CF_CFDP_ResetTransaction().

Referenced by CF_AbandonCmd().

Here is the call graph for this function:



12.38.4.2 CF_AbandonCmd() CFE_Status_t CF_AbandonCmd (

```
const CF_AbandonCmd_t * msg )
```

Handle an abandon ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

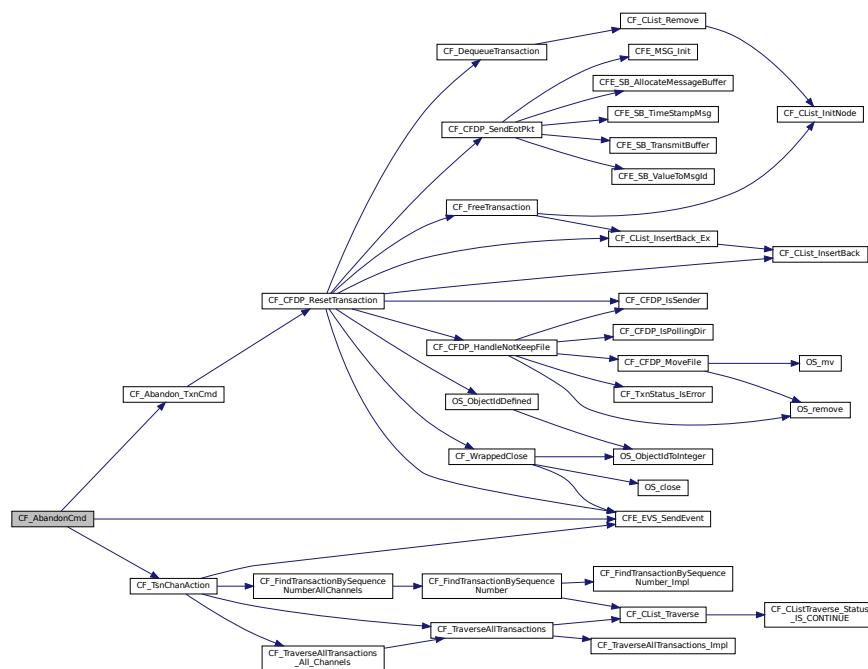
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 537 of file cf_cmd.c.

References CF_Abandon_TxnCmd(), CF_AppData, CF_CMD_ABANDON_CHAN_ERR_EID, CF_CMD_ABANDON_INF_EID, CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CF_E_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_AbandonCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



```

12.38.4.3 CF_Cancel_TxnCmd() void CF_Cancel_TxnCmd (
    CF_Transaction_t * txn,
    void * ignored )
  
```

tsn chan action to cancel a transaction.

This helper function is used with [CF_TsnChanAction\(\)](#) to cancel matched transactions

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

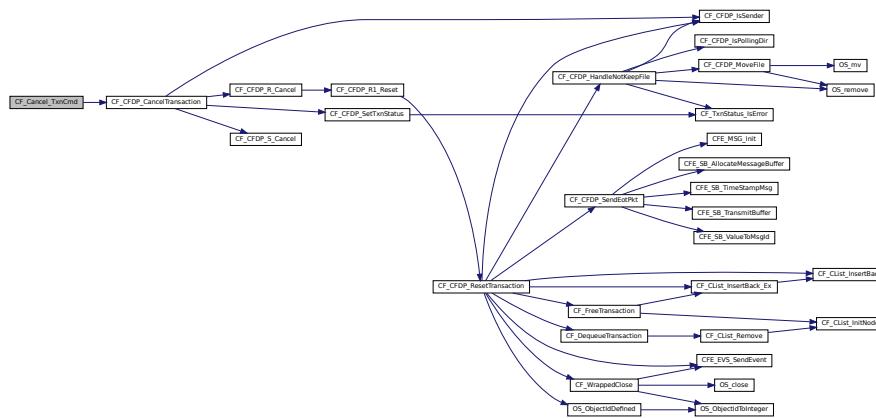
<code>txn</code>	Pointer to transaction object
<code>ignored</code>	Not used by this function

Definition at line 491 of file cf_cmd.c.

References CF_CFDP_CancelTransaction().

Referenced by CF_CancelCmd().

Here is the call graph for this function:



12.38.4.4 CF_CancelCmd() `CFE_Status_t CF_CancelCmd (const CF_CancelCmd_t * msg)`

Handle a cancel ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

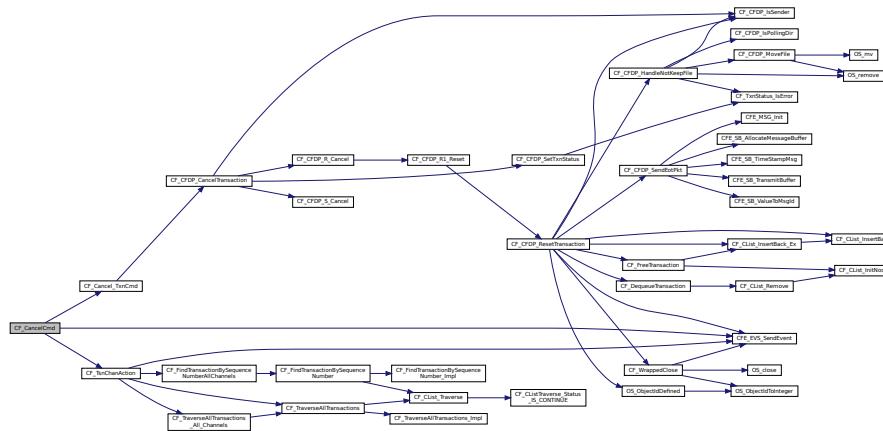
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 502 of file cf_cmd.c.

References CF_AppData, CF_Cancel_TxnCmd(), CF_CMD_CANCEL_CHAN_ERR_EID, CF_CMD_CANCEL_IN←F_EID, CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS←SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters←::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_CancelCmd::Payload.

Here is the call graph for this function:



12.38.4.5 CF_ChanAction_Status_IS_SUCCESS() static bool CF_ChanAction_Status_IS_SUCCESS (CF_ChanAction_Status_t stat) [inline], [static]

Checks if the channel action was successful

Definition at line 42 of file cf_cmd.h.

References CF_ChanAction_Status_SUCCESS.

Referenced by CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_FreezeCmd(), CF_GetSetParamCmd(), CF_PurgeQueueCmd(), and CF_ThawCmd().

12.38.4.6 CF_DisableDequeueCmd() CFE_Status_t CF_DisableDequeueCmd (const CF_DisableDequeueCmd_t * msg)

Handle a disable dequeue ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

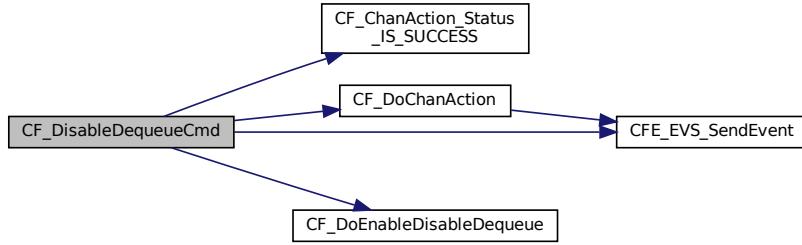
Parameters

msg	Pointer to command message
-----	----------------------------

Definition at line 600 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_DISABLE_DEQUEUE_ERR_EID, CF_CMD_DISABLE_DEQUEUE_INF_EID, CF_DoChanAction(), CF_DoEnableDisableDequeue(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_DisableDequeueCmd::Payload.

Here is the call graph for this function:



12.38.4.7 CF_DisableDirPollingCmd() `CFE_Status_t CF_DisableDirPollingCmd (const CF_DisableDirPollingCmd_t * msg)`

Disable a polling dir ground command.

Assumptions, External Events, and Notes:

`msg` must not be NULL.

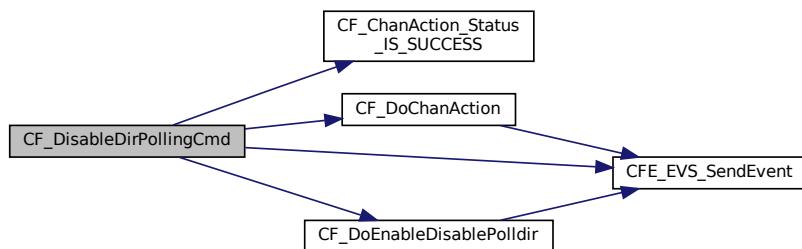
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 685 of file cf_cmd.c.

References `CF_AppData`, `CF_ChanAction_Status_IS_SUCCESS()`, `CF_CMD_DISABLE_POLLDIR_ERR_EID`, `CF_CMD_DISABLE_POLLDIR_INF_EID`, `CF_DoChanAction()`, `CF_DoEnableDisablePolldir()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_HkCmdCounters::cmd`, `CF_HkPacket_Payload::counters`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_DisableDirPollingCmd::Payload`.

Here is the call graph for this function:



12.38.4.8 CF_DisableEngineCmd() `CFE_Status_t CF_DisableEngineCmd (const CF_DisableEngineCmd_t * msg)`

Ground command disable engine.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

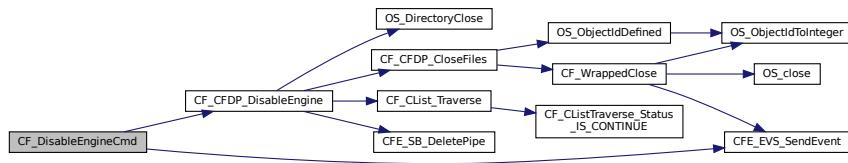
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1201 of file cf_cmd.c.

References CF_AppData, CF_CFDP_DisableEngine(), CF_CMD_DISABLE_ENGINE_INF_EID, CF_CMD_ENG_A_LREADY_DIS_INF_EID, CFE_EVT_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_Hk_CmdCounters::cmd, CF_HkPacket_Payload::counters, CF_Engine::enabled, CF_AppData_t::engine, CF_AppData_t::hk, and CF_HkPacket::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.38.4.9 CF_DoChanAction() `CF_ChanAction_Status_t CF_DoChanAction (`

```

    const CF_UnionArgs_Payload_t * data,
    const char * errstr,
    CF_ChanActionFn_t fn,
    void * context )
  
```

Common logic for all channel-based commands.

Description

All the commands that act on channels or have the special "all channels" parameter come through this function. This puts all common logic in one place. It does not handle the command accept or reject counters.

Assumptions, External Events, and Notes:

cmd must not be NULL, errstr must not be NULL, fn must be a valid function, context may be NULL.

Parameters

<code>data</code>	Pointer to payload being processed
<code>errstr</code>	String to be included in the EVS event if command should fail
<code>fn</code>	Callback action function to invoke for each affected channel
<code>context</code>	Opaque pointer to pass through to callback (not used in this function)

Returns

The return value from the given action function.

Return values

<i>CF_ChanAction_Status_ERROR</i>	on error
-----------------------------------	----------

Definition at line 233 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_ALL_CHANNELS, CF_ChanAction_Status_ERROR, CF_ChanAction_Status_SUCCESS, CF_CMD_CHAN_PARAM_ERR_EID, CF_NUM_CHANNELS, CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by CF_DisableDequeueCmd(), CF_DisableDirPollingCmd(), CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_FreezeCmd(), CF_PurgeQueueCmd(), and CF_ThawCmd().

Here is the call graph for this function:



12.38.4.10 CF_DoEnableDisableDequeue() *CF_ChanAction_Status_t* CF_DoEnableDisableDequeue (
uint8 chan_num,
*void * arg*)

Sets the dequeue enable/disable flag for a channel.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be <i>CF_ChanAction_BoolArg_t*</i>

Returns

Always succeeds, so returns CF_ChanAction_Status_SUCCESS.

Definition at line 561 of file cf_cmd.c.

References CF_ChanAction_BoolArg::barg, CF_AppData, CF_ChanAction_Status_SUCCESS, CF_ConfigTable::chan, CF_AppData_t::config_table, and CF_ChannelConfig::dequeue_enabled.

Referenced by CF_DisableDequeueCmd(), and CF_EnableDequeueCmd().

12.38.4.11 CF_DoEnableDisablePolldir() *CF_ChanAction_Status_t* CF_DoEnableDisablePolldir (

```
    uint8 chan_num,
    void * arg )
```

Sets the enable/disable flag for the specified polling directory.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChанAction_BoolMsgArg_t*

Returns

success/fail status code

Return values

<i>CF_ChанAction_Status_SUCCESS</i>	if successful
<i>CF_ChанAction_Status_ERROR</i>	if failed

Definition at line 625 of file cf_cmd.c.

References CF_ChанAction_BoolMsgArg::barg, CF_UnionArgs_Payload::byte, CF_ALL_POLLDIRS, CF_AppData, CF_ChанAction_Status_ERROR, CF_ChанAction_Status_SUCCESS, CF_CMD_POLLDIR_INVALID_ERR_EID, C← F_MAX_POLLING_DIR_PER_CHAN, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_ConfigTable← ::chan, CF_AppData_t::config_table, CF_ChанAction_BoolMsgArg::data, CF_PollDir::enabled, and CF_Channel← Config::polldir.

Referenced by CF_DisableDirPollingCmd(), and CF_EnableDirPollingCmd().

Here is the call graph for this function:



12.38.4.12 CF_DoFreezeThaw() CF_ChанAction_Status_t CF_DoFreezeThaw (

```
    uint8 chan_num,
    void * arg )
```

Channel action to set the frozen bit for a channel.

Assumptions, External Events, and Notes:

context must not be NULL.

Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChанAction_BoolArg_t*

Returns

Always succeeds, so returns CF_ChанAction_Status_SUCCESS.

Definition at line 269 of file cf_cmd.c.

References CF_ChанAction_BoolArg::barg, CF_AppData, CF_ChанAction_Status_SUCCESS, CF_HkPacket<→Payload::channel_hk, CF_HkChannel_Data::frozen, CF_AppData_t::hk, and CF_HkPacket::Payload.
Referenced by CF_FreezeCmd(), and CF_ThawCmd().

12.38.4.13 CF_DoPurgeQueue() CF_ChанAction_Status_t CF_DoPurgeQueue (

```
    uint8 chan_num,
    void * arg )
```

Channel action command to perform purge queue operations.

Description

Determines from the command parameters which queues to traverse and purge state.

Assumptions, External Events, and Notes:

None

Parameters

<i>chan_num</i>	CF channel number
<i>arg</i>	Pointer to purge queue command

Returns

integer status code indicating success or failure

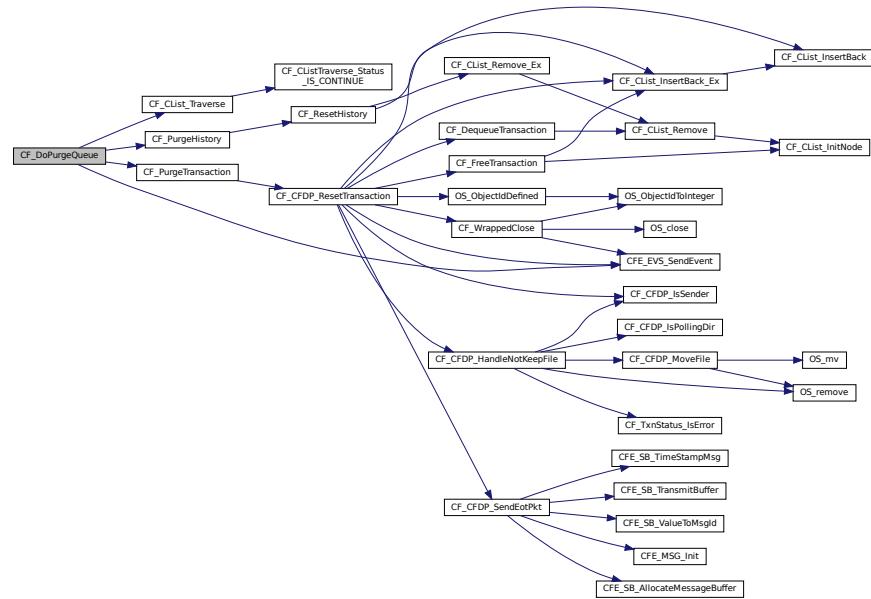
Return values

<i>CF_ChанAction_Status_SUCCESS</i>	if successful
<i>CF_ChанAction_Status_ERROR</i>	on error

Definition at line 739 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_AppData, CF_ChанAction_Status_ERROR, CF_ChанAction_Status<→SUCCESS, CF_CList_Traverse(), CF_CMD_PURGE_ARG_ERR_EID, CF_PurgeHistory(), CF_PurgeTransaction(), CF_QueueIdx_HIST, CF_QueueIdx_PEND, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CF_Engine<→::channels, CF_AppData_t::engine, and CF_Channel::qs.
Referenced by CF_PurgeQueueCmd().

Here is the call graph for this function:



```

12.38.4.14 CF_DoSuspRes() void CF_DoSuspRes (
    const CFE_Transaction_Payload_t * payload,
    uint8 action )
  
```

Handle transaction suspend and resume commands.

Description

This is called for both suspend and resume ground commands. It uses the [CF_TsnChanAction\(\)](#) function to perform the command.

Assumptions, External Events, and Notes:

payload must not be NULL.

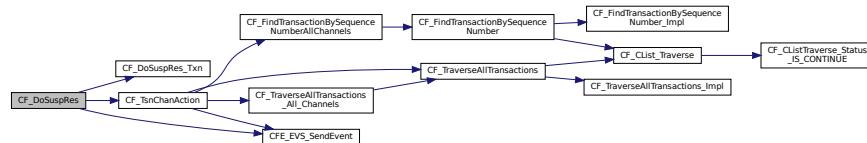
Parameters

<i>payload</i>	Pointer to the command message
<i>action</i>	Action to take (suspend or resume)

Definition at line 426 of file cf_cmd.c.

References CF_AppData, CF_CMD_SUSPRES_CHAN_ERR_EID, CF_CMD_SUSPRES_INF_EID, CF_CMD_SUSPRES_SAME_INF_EID, CF_DoSuspRes_Txn(), CF_TsnChanAction(), CFE_EVS_EventType_ERROR, CFE_EVT_INFORMATION, CFE_EVS_SendEvent(), CF_HkCmdCounters::cmd, CF_HkPacket::Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_ChanAction_SuspResArg::same. Referenced by CF_ResumeCmd(), and CF_SuspendCmd().

Here is the call graph for this function:



12.38.4.15 CF_DoSuspRes_Txn() `void CF_DoSuspRes_Txn (`
`CF_Transaction_t * txn,`
`CF_ChanAction_SuspResArg_t * context)`

Set the suspended bit in a transaction.

Assumptions, External Events, and Notes:

txn must not be NULL. context must not be NULL.

Parameters

<i>txn</i>	Pointer to the transaction object
<i>context</i>	Pointer to CF_ChanAction_SuspResArg_t structure from initial call

Definition at line 407 of file cf_cmd.c.

References CF_ChanAction_SuspResArg::action, CF_Assert, CF_StateFlags::com, CF_Transaction::flags, CF_ChanAction_SuspResArg::same, and CF_Flags_Common::suspended.

Referenced by CF_DoSuspRes().

12.38.4.16 CF_EnableDequeueCmd() `CFE_Status_t CF_EnableDequeueCmd (`
`const CF_EnableDequeueCmd_t * msg)`

Handle an enable dequeue ground command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

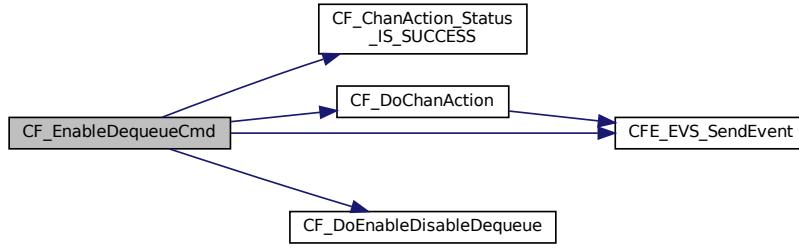
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 575 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_ENABLE_DEQUEUE_ERR_EID, CF_CMD_ENABLE_DEQUEUE_INF_EID, CF_DoChanAction(), CF_DoEnableDisableDequeue(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_EnableDequeueCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.38.4.17 `CF_EnableDirPollingCmd()` `CFE_Status_t CF_EnableDirPollingCmd (`
`const CF_EnableDirPollingCmd_t * msg)`

Enable a polling dir ground command.

Assumptions, External Events, and Notes:

`msg` must not be NULL.

Parameters

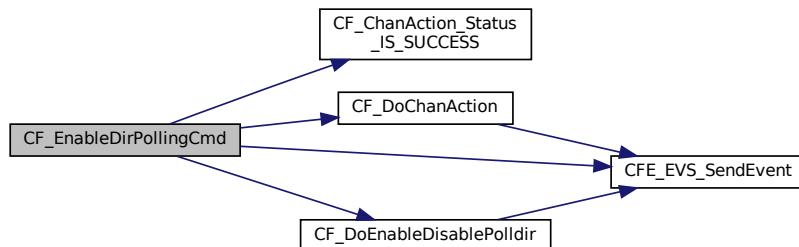
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 658 of file `cf_cmd.c`.

References `CF_AppData`, `CF_ChanAction_Status_IS_SUCCESS()`, `CF_CMD_ENABLE_POLLDIR_ERR_EID`, `CF_CMD_ENABLE_POLLDIR_INF_EID`, `CF_DoChanAction()`, `CF_DoEnableDisablePolldir()`, `CFE_EVS_EventType_EROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_HkCmdCounters::cmd`, `CF_HkPacket_Payload::counters`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_EnableDirPollingCmd::Payload`.

Referenced by `CF_ProcessGroundCommand()`.

Here is the call graph for this function:



12.38.4.18 CF_EnableEngineCmd() `CFE_Status_t CF_EnableEngineCmd (const CF_EnableEngineCmd_t * msg)`

Ground command enable engine.

Assumptions, External Events, and Notes:

msg must not be NULL.

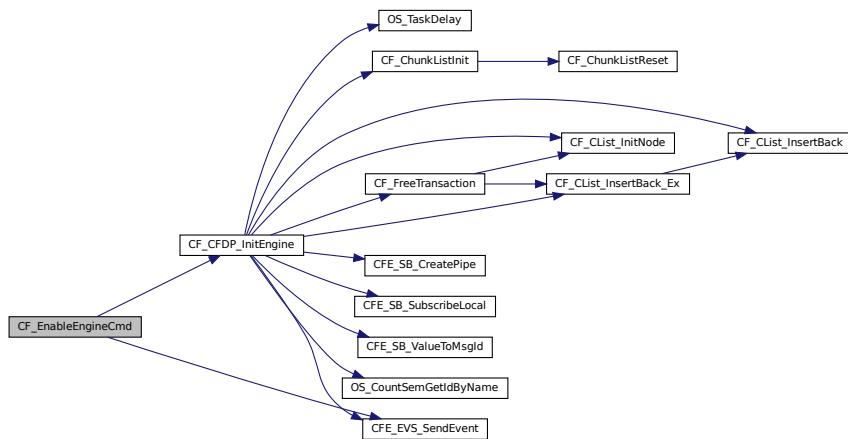
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1169 of file cf_cmd.c.

References CF_AppData, CF_CFDP_InitEngine(), CF_CMD_ENABLE_ENGINE_ERR_EID, CF_CMD_ENABLE_ENGINE_INF_EID, CF_CMD_ENG_ALREADY_ENA_INF_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_Engine::enabled, CF_AppData_t::engine, CF_HkCmdCounters::err, CF_AppData_t::hk, and CF_HkPacket::Payload.

Here is the call graph for this function:



12.38.4.19 CF_FindTransactionBySequenceNumberAllChannels() `CF_Transaction_t* CF_FindTransactionBySequenceNumberAllChannels (CF_TransactionSeq_t ts, CF_EntityId_t eid)`

Search for a transaction across all channels.

Assumptions, External Events, and Notes:

None

Parameters

<code>ts</code>	Transaction sequence number to find
<code>eid</code>	Entity ID of the transaction

Returns

Pointer to the transaction, if found

Return values

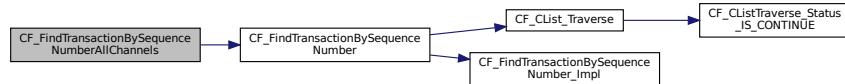
<code>NULL</code>	if transaction not found
-------------------	--------------------------

Definition at line 331 of file cf_cmd.c.

References CF_AppData, CF_FindTransactionBySequenceNumber(), CF_NUM_CHANNELS, CF_Engine::channels, and CF_AppData_t::engine.

Referenced by CF_TsnChanAction().

Here is the call graph for this function:



12.38.4.20 CF_FreezeCmd() `CFE_Status_t CF_FreezeCmd (`
`const CF_FreezeCmd_t * msg)`

Freeze a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

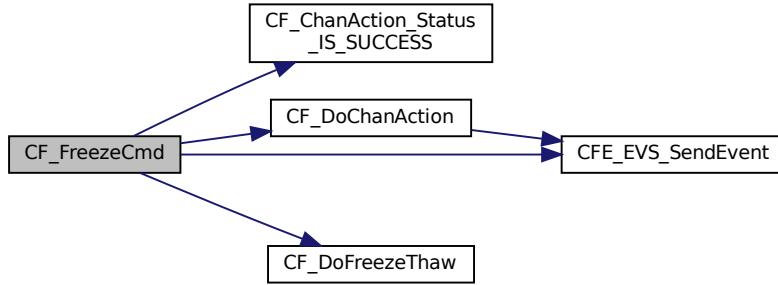
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 283 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_FREEZE_ERR_EID, CF_CMD_FREEZE_INF_EID, CF_DoChanAction(), CF_DoFreezeThaw(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_FreezeCmd::Payload.

Here is the call graph for this function:



12.38.4.21 CF_GetParamCmd() `CFE_Status_t CF_GetParamCmd (`
`const CF_GetParamCmd_t * msg)`

Ground command to set a configuration parameter.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

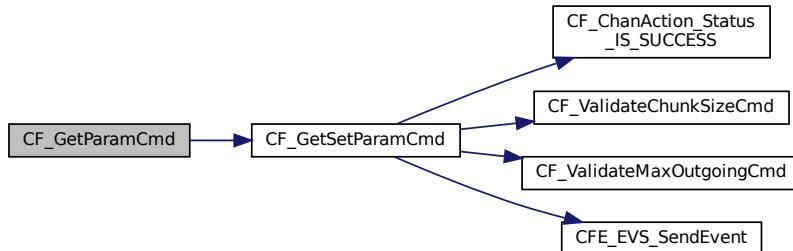
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1154 of file cf_cmd.c.

References CF_SetParamCmd(), CFE_SUCCESS, CF_GetParam_Payload::chan_num, CF_GetParam_Payload::key, and CF_GetParamCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



```
12.38.4.22 CF_GetSetParamCmd() void CF_GetSetParamCmd (
    bool is_set,
    CF_GetSet_ValueID_t param_id,
    uint32 value,
    uint8 chan_num )
```

Perform a configuration get/set operation.

For a set, this sets the value within the CF configuration. For a get, this generates an EVS event with the requested information.

Description

Combine get and set in one function with common logic.

Assumptions, External Events, and Notes:

None

Parameters

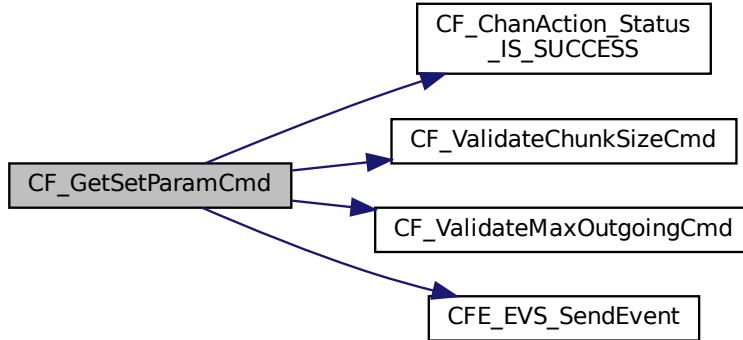
<i>is_set</i>	Whether to get (false) or set (true)
<i>param_id</i>	Parameter ID
<i>value</i>	Value to get/set
<i>chan_num</i>	Channel number to operate on

Definition at line 984 of file cf_cmd.c.

References CF_ChannelConfig::ack_limit, CF_ChannelConfig::ack_timer_s, CF_AppData, CF_ChAction_Status_IS_SUCCESS(), CF_CMD_GETSET1_INF_EID, CF_CMD_GETSET2_INF_EID, CF_CMD_GETSET_CHAN_ERR_EID, CF_CMD_GETSET_PARAM_ERR_EID, CF_CMD_GETSET_VALIDATE_ERR_EID, CF_ERROR, CF_GetSet_ValueID_ack_limit, CF_GetSet_ValueID_ack_timer_s, CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup, CF_GetSet_ValueID_inactivity_timer_s, CF_GetSet_ValueID_local_eid, CF_GetSet_ValueID_nak_limit, CF_GetSet_ValueID_nak_timer_s, CF_GetSet_ValueID_outgoing_file_chunk_size, CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup, CF_GetSet_ValueID_ticks_per_second, CF_NUM_CHANNELS, CF_ValidateChunkSizeCmd(), CF_ValidateMaxOutgoingCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_ConfigTable::chan, CF_HkCmdCounters::cmd, CF_AppData_t::config_table, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_ChannelConfig::inactivity_timer_s, CF_ConfigTable::local_eid, CF_ChannelConfig::max_outgoing_messages_per_wakeup, CF_ChannelConfig::nak_limit, CF_ChannelConfig::nak_timer_s, CF_ConfigTable::outgoing_file_chunk_size, CF_HkPacket::Payload, CF_ConfigTable::rx_crc_calc_bytes_per_wakeup, and CF_ConfigTable::ticks_per_second.

Referenced by CF_GetParamCmd(), and CF_SetParamCmd().

Here is the call graph for this function:



12.38.4.23 CF_NoopCmd() `CFE_Status_t CF_NoopCmd (`
`const CF_NoopCmd_t * msg)`

The no-operation command.

Description

This function has a signature the same of all cmd_ functions. This function simply prints an event message. Increments the command accept counter. The msg parameter is ignored in this one.

Assumptions, External Events, and Notes:

None

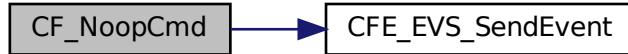
Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 48 of file cf_cmd.c.

References CF_AppData, CF_MAJOR_VERSION, CF_MINOR_VERSION, CF_MISSION_REV, CF_NOOP_INF_E_ID, CF_REVISION, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_AppData_t::hk, and CF_HkPacket::Payload.

Here is the call graph for this function:



12.38.4.24 CF_PlaybackDirCmd() [CF_Status_t](#) CF_PlaybackDirCmd (const CF_PlaybackDirCmd_t * msg)

Ground command to start directory playback.

Description

This function has a signature the same of all cmd_ functions. Increments the command accept or reject counter.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

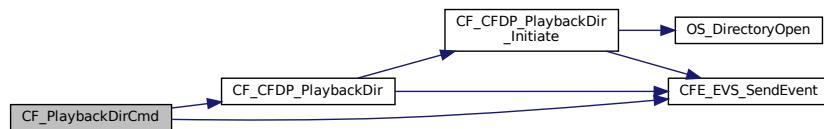
msg	Pointer to command message
-----	----------------------------

Definition at line 183 of file cf_cmd.c.

References CF_AppData, CF_CFDP_CLASS_1, CF_CFDP_CLASS_2, CF_CFDP_PlaybackDir(), CF_CMD_BAD_← PARAM_ERR_EID, CF_CMD_PLAYBACK_DIR_ERR_EID, CF_CMD_PLAYBACK_DIR_INF_EID, CF_NUM_CHAN← NELS, CF_TxFile_Payload::cfdp_class, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CF← E_EVS_SendEvent(), CFE_SUCCESS, CF_TxFile_Payload::chan_num, CF_HkCmdCounters::cmd, CF_HkPacket← _Payload::counters, CF_TxFile_Payload::dest_id, CF_TxFile_Payload::dst_filename, CF_HkCmdCounters::err, C← F_AppData_t::hk, CF_TxFile_Payload::keep, CF_HkPacket::Payload, CF_PlaybackDirCmd::Payload, CF_TxFile← Payload::priority, and CF_TxFile_Payload::src_filename.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.38.4.25 CF_PurgeHistory() [CF_CListTraverse_Status_t](#) CF_PurgeHistory (

```
CF_CListNode_t * node,
void * arg )
```

Purge the history queue for the given channel.

This helper function is used in conjunction with [CF_CList_Traverse\(\)](#)

Assumptions, External Events, and Notes:

node must not be NULL. chan must not be NULL.

Parameters

<i>node</i>	Current list node being traversed
<i>arg</i>	Channel pointer passed through as opaque object, must be CF_Channel_t*

Returns

Always [CF_CLIST_CONT](#) to process all entries

Definition at line 712 of file cf_cmd.c.

References [CF_CLIST_CONT](#), [CF_ResetHistory\(\)](#), and [container_of](#).

Referenced by [CF_DoPurgeQueue\(\)](#).

Here is the call graph for this function:



12.38.4.26 CF_PurgeQueueCmd() CFE_Status_t CF_PurgeQueueCmd (

```
const CF_PurgeQueueCmd_t * msg )
```

Ground command to purge either the history or pending queues.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

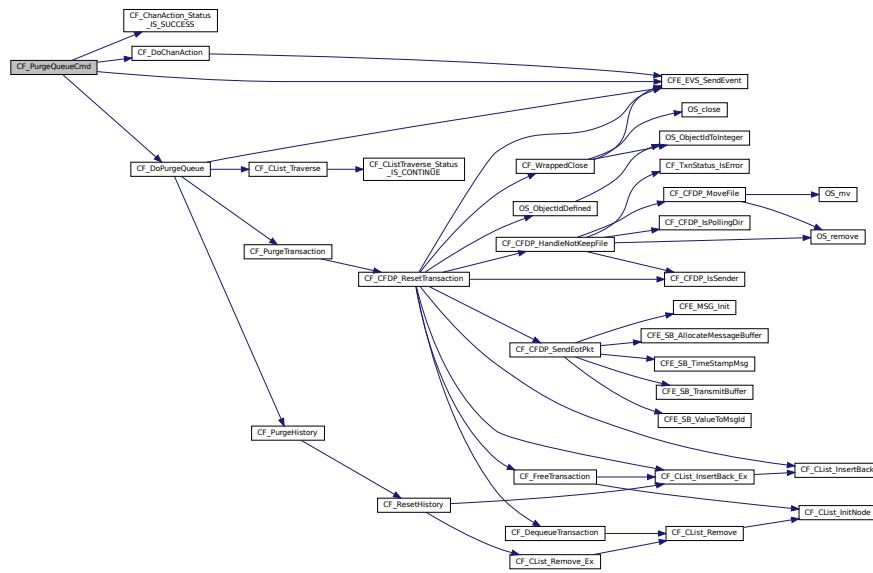
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 790 of file cf_cmd.c.

References [CF_AppData](#), [CF_ChAction_Status_IS_SUCCESS\(\)](#), [CF_CMD_PURGE_QUEUE_ERR_EID](#), [CF_CMD_PURGE_QUEUE_INF_EID](#), [CF_DoChanAction\(\)](#), [CF_DoPurgeQueue\(\)](#), [CFE_EVS_EventType_ERROR](#), [CFE_EVS_EventType_INFORMATION](#), [CFE_EVS_SendEvent\(\)](#), [CFE_SUCCESS](#), [CF_HkCmdCounters::cmd](#), [CF_HkPacket_Payload::counters](#), [CF_HkCmdCounters::err](#), [CF_AppData_t::hk](#), [CF_HkPacket::Payload](#), and [CF_PurgeQueueCmd::Payload](#).

Referenced by [CF_ProcessGroundCommand\(\)](#).

Here is the call graph for this function:



```
12.38.4.27 CF_PurgeTransaction() CF_CListTraverse_Status_t CF_PurgeTransaction (
```

<pre> CF_CListNode_t * node,</pre>	<pre> void * ignored)</pre>
---------------------------------------	---------------------------------

Purge the pending transaction queue.

This helper function is used in conjunction with [CF_CList_Traverse\(\)](#)

Assumptions, External Events, and Notes:

node must not be NULL.

Parameters

<i>node</i>	Current list node being traversed
<i>ignored</i>	Not used by this implementation

Returns

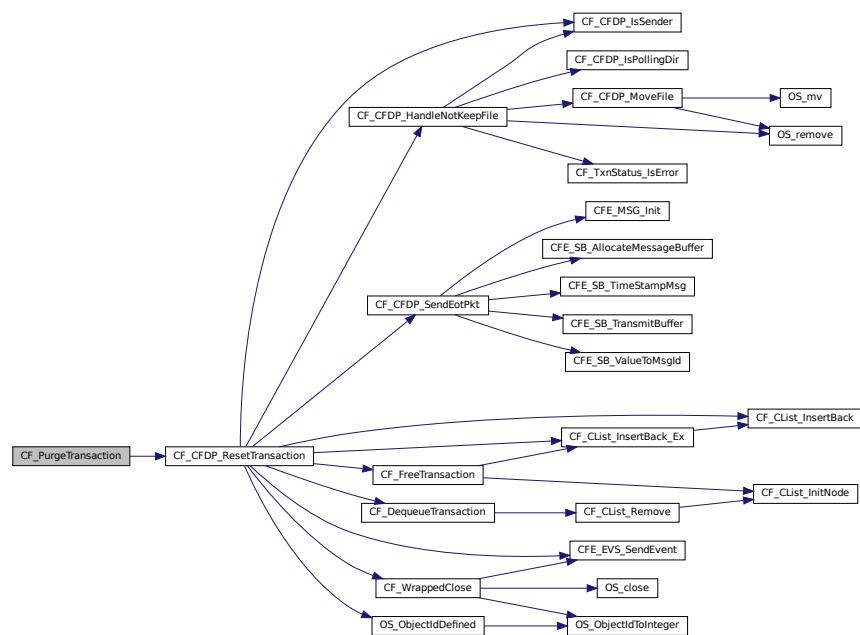
Always [CF_CLIST_CONT](#) to process all entries

Definition at line 726 of file cf_cmd.c.

References CF_CFDP_ResetTransaction(), CF_CLIST_CONT, and container_of.

Referenced by CF_DoPurgeQueue().

Here is the call graph for this function:



12.38.4.28 CF_ResetCountersCmd() [CFE_Status_t](#) CF_ResetCountersCmd (const [CF_ResetCountersCmd_t](#) * msg)

The reset counters command.

Description

This function has a signature the same of all cmd_ functions. Resets the given counter, or all. Increments the command accept or reject counter. If the command counters are reset, then there is no increment.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 64 of file cf_cmd.c.

References CF_UnionArgs_Payload::byte, CF_AppData, CF_CMD_RESET_INVALID_ERR_EID, CF_NUM_CHAN< NELS, CF_Reset_all, CF_Reset_command, CF_Reset_down, CF_Reset_fault, CF_RESET_INF_EID, CF_Reset<=

up, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkPacket_Payload::channel_hk, CF_HkCmdCounters::cmd, CF_HkChannel_Data::counters, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_HkCounters::fault, CF_AppData_t::hk, CF_HkPacket::Payload, CF_ResetCountersCmd::Payload, CF_HkCounters::recv, and CF_HkCounters::sent.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.38.4.29 CF_ResumeCmd() `CFE_Status_t CF_ResumeCmd (const CF_ResumeCmd_t * msg)`

Handle transaction resume command.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

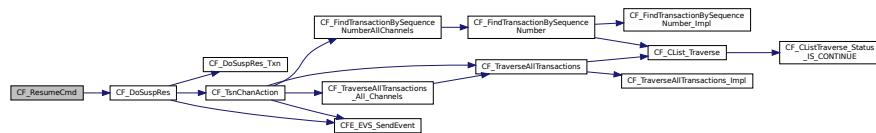
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 479 of file cf_cmd.c.

References CF_DoSuspRes(), CFE_SUCCESS, and CF_ResumeCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



12.38.4.30 CF_SendHkCmd() `CFE_Status_t CF_SendHkCmd (const CF_SendHkCmd_t * msg)`

Send CF housekeeping packet.

Description

The command to send the CF housekeeping packet

Assumptions, External Events, and Notes:

None

Parameters

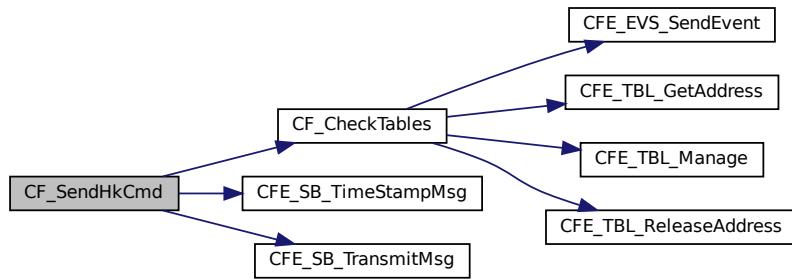
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1225 of file cf_cmd.c.

References CF_AppData, CF_CheckTables(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SUCCESS, CF_AppData_t::hk, and CF_HkPacket::TelemetryHeader.

Referenced by CF_AppPipe().

Here is the call graph for this function:

**12.38.4.31 CF_SetParamCmd()**

```
CFE_Status_t CF_SetParamCmd(
```

```
    const CF_SetParamCmd_t * msg )
```

Ground command to set a configuration parameter.

Assumptions, External Events, and Notes:

`msg` must not be NULL.

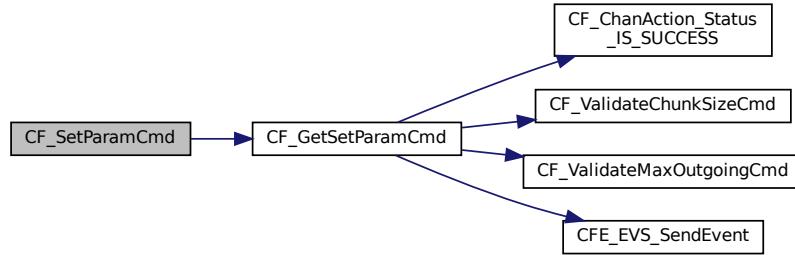
Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1139 of file cf_cmd.c.

References CF_GetSetParamCmd(), CFE_SUCCESS, CF_SetParam_Payload::chan_num, CF_SetParam_Payload::key, CF_SetParamCmd::Payload, and CF_SetParam_Payload::value.

Here is the call graph for this function:



12.38.4.32 CF_SuspendCmd() `CFE_Status_t CF_SuspendCmd (const CF_SuspendCmd_t * msg)`

Handle transaction suspend command.

Assumptions, External Events, and Notes:

msg must not be NULL.

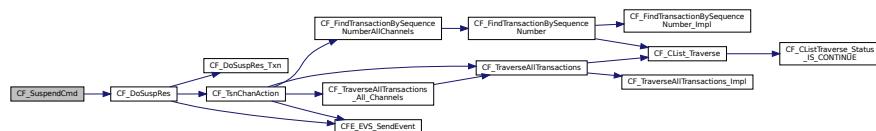
Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 467 of file cf_cmd.c.

References CF_DoSuspRes(), CFE_SUCCESS, and CF_SuspendCmd::Payload.

Here is the call graph for this function:



12.38.4.33 CF_ThawCmd() `CFE_Status_t CF_ThawCmd (`
 `const CF_ThawCmd_t * msg)`

Thaw a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

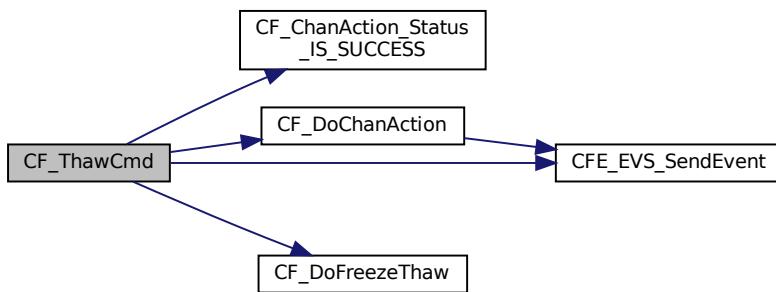
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 307 of file cf_cmd.c.

References CF_AppData, CF_ChanAction_Status_IS_SUCCESS(), CF_CMD_THAW_ERR_EID, CF_CMD_THAW_INF_EID, CF_DoChanAction(), CF_DoFreezeThaw(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_HkPacket::Payload, and CF_ThawCmd::Payload.

Referenced by CF_ProcessGroundCommand().

Here is the call graph for this function:



```

12.38.4.34 CF_TsnChanAction() int32 CF_TsnChanAction (
    const CF_Transaction_Payload_t * data,
    const char * cmdstr,
    CF_TsnChanAction_fn_t fn,
    void * context )
  
```

Common logic for all transaction sequence number and channel commands.

Description

All the commands that on a transaction on a particular channel come through this function. This puts all common logic in one place. It does handle the command accept or reject counters.

Assumptions, External Events, and Notes:

cmd must not be NULL, fn must be a valid function, context may be NULL.

Parameters

<i>data</i>	Pointer to payload being processed
<i>cmdstr</i>	String to include in any generated EVS events
<i>fn</i>	Callback function to invoke for each matched transaction
<i>context</i>	Opaque object to pass through to the callback

Returns

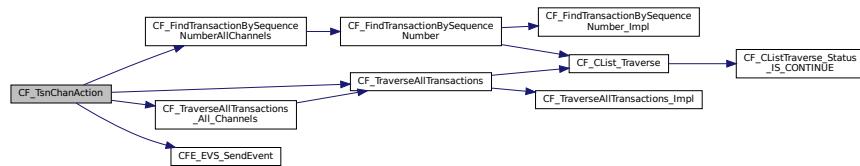
returns the number of transactions acted upon

Definition at line 359 of file cf_cmd.c.

References CF_ALL_CHANNELS, CF_AppData, CF_CMD_TRANS_NOT_FOUND_ERR_EID, CF_CMD_TSN_CH_AN_INVALID_ERR_EID, CF_COMPOUND_KEY, CF_FindTransactionBySequenceNumberAllChannels(), CF_NUM_CHANNELS, CF_TraverseAllTransactions(), CF_TraverseAllTransactions_All_Channels(), CFE_EVS_EventType_ER_ROR, CFE_EVS_SendEvent(), CF_Transaction_Payload::chan, CF_Engine::channels, CF_Transaction_Payload::eid, CF_AppData_t::engine, and CF_Transaction_Payload::ts.

Referenced by CF_AbandonCmd(), CF_CancelCmd(), and CF_DoSuspRes().

Here is the call graph for this function:



12.38.4.35 CF_TxFileCmd() CFE_Status_t CF_TxFileCmd (

const CF_TxFileCmd_t * msg)

Ground command to start a file transfer.

Description

This function has a signature the same of all cmd_ functions. Increments the command accept or reject counter.

Assumptions, External Events, and Notes:

msg must not be NULL.

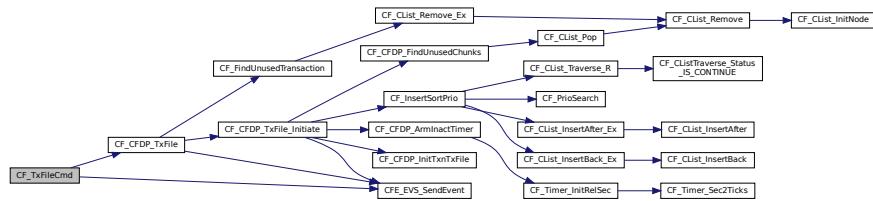
Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 134 of file cf_cmd.c.

References CF_AppData, CF_CFDP_CLASS_1, CF_CFDP_CLASS_2, CF_CFDP_TxFile(), CF_CMD_BAD_PARAMETER_ERR_EID, CF_CMD_TX_FILE_ERR_EID, CF_CMD_TX_FILE_INF_EID, CF_NUM_CHANNELS, CF_TxFile_Payload::cfdp_class, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_TxFile_Payload::chan_num, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_TxFile_Payload::dest_id, CF_TxFile_Payload::dst_filename, CF_HkCmdCounters::err, CF_AppData_t::hk, CF_TxFile_Payload::keep, CF_HkPacket::Payload, CF_TxFileCmd::Payload, CF_TxFile_Payload::priority, and CF_TxFile_Payload::src_filename.

Here is the call graph for this function:



12.38.4.36 CF_ValidateChunkSizeCmd() `CF_ChanAction_Status_t CF_ValidateChunkSizeCmd (CF_ChunkSize_t val, uint8 chan_num)`

Checks if the value is less than or equal to the max PDU size.

Assumptions, External Events, and Notes:

None

Parameters

<code>val</code>	Size of chunk to test
<code>chan_num</code>	Ignored by this implementation

Returns

status code indicating if check passed

Return values

<code>CF_ChanAction_Status_SUCCESS</code>	if successful (val is less than or equal to max PDU)
<code>CF_ChanAction_Status_ERROR</code>	if failed (val is greater than max PDU)

Definition at line 949 of file cf_cmd.c.

References `CF_ChanAction_Status_ERROR`, and `CF_ChanAction_Status_SUCCESS`.

Referenced by `CF_GetSetParamCmd()`.

12.38.4.37 CF_ValidateMaxOutgoingCmd() `CF_ChanAction_Status_t CF_ValidateMaxOutgoingCmd (uint32 val, uint8 chan_num)`

Checks if the value is within allowable range as outgoing packets per wakeup.

Assumptions, External Events, and Notes:

None

Parameters

<i>val</i>	Number to test
<i>chan_num</i>	CF channel number

Returns

status code indicating if check passed

Return values

<i>CF_ChanAction_Status_SUCCESS</i>	if successful (val is allowable as max packets per wakeup)
<i>CF_ChanAction_Status_ERROR</i>	if failed (val is not allowed)

Definition at line 965 of file cf_cmd.c.

References CF_AppData, CF_ChанAction_Status_ERROR, CF_ChанAction_Status_SUCCESS, CF_ConfigTable<→
::chan, CF_AppData_t::config_table, and CF_ChannelConfig::sem_name.

Referenced by CF_GetSetParamCmd().

12.38.4.38 CF_WakeupCmd() *CFE_Status_t* CF_WakeupCmd (

```
    const CF_WakeupCmd_t * msg )
```

CF wakeup function.

Description

Performs a single engine cycle for each wakeup

Assumptions, External Events, and Notes:

None

Parameters

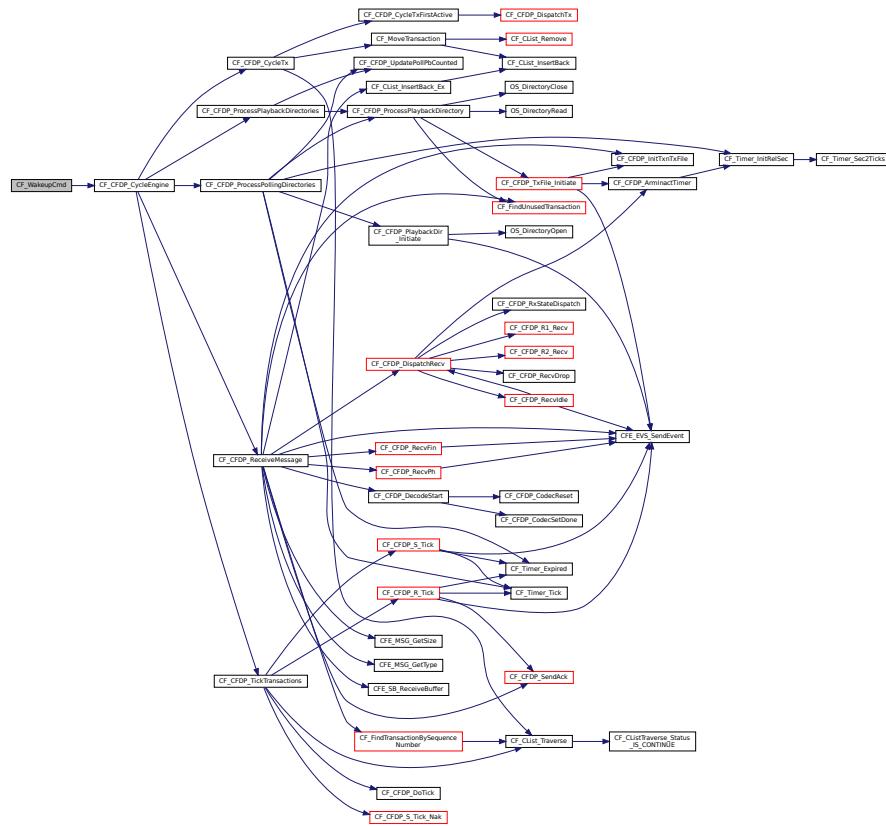
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1242 of file cf_cmd.c.

References CF_CFDP_CycleEngine(), CF_PERF_ID_CYCLE_ENG, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit,
and CFE_SUCCESS.

Referenced by CF_AppPipe().

Here is the call graph for this function:



12.38.4.39 CF_WriteQueueCmd() `CFE_Status_t CF_WriteQueueCmd (`
`const CF_WriteQueueCmd_t * msg)`

Ground command to write a file with queue information.

Assumptions, External Events, and Notes:

msg must not be NULL.

Parameters

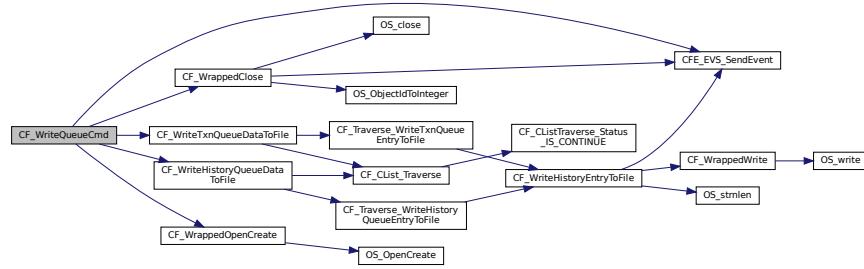
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 813 of file cf_cmd.c.

References `CF_AppData`, `CF_CMD_WQ_ARGS_ERR_EID`, `CF_CMD_WQ_CHAN_ERR_EID`, `CF_CMD_WQ_INF_EID`, `CF_CMD_WQ_OPEN_ERR_EID`, `CF_CMD_WQ_WRITEHIST_RX_ERR_EID`, `CF_CMD_WQ_WRITEHIST_TX_ERR_EID`, `CF_CMD_WQ_WRITEQ_PEND_ERR_EID`, `CF_CMD_WQ_WRITEQ_RX_ERR_EID`, `CF_CMD_WQ_WRITEQ_RX_ERR_EID`, `CF_DIRECTION_RX`, `CF_DIRECTION_TX`, `CF_NUM_CHANNELS`, `CF_Queue_active`, `CF_Queue_all`, `CF_Queue_history`, `CF_Queue_pend`, `CF_QueueIdx_PEND`, `CF_QueueIdx_RX`, `CF_QueueIdx_TXA`, `CF_Queue_Idx_TXW`, `CF_Type_all`, `CF_Type_down`, `CF_Type_up`, `CF_WrappedClose()`, `CF_WrappedOpenCreate()`, `CF_WriteHistoryQueueDataToFile()`, `CF_WriteTxnQueueDataToFile()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType`.

_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CF_WriteQueue_Payload::chan, CF_Engine::channels, CF_HkCmdCounters::cmd, CF_HkPacket_Payload::counters, CF_AppData_t::engine, CF_HkCmdCounters::err, CF_WriteQueue_Payload::filename, CF_AppData_t::hk, OS_FILE_FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_OBJECT_ID_UNDEFINED, OS_WRITE_ONLY, CF_HkPacket::Payload, CF_WriteQueueCmd::Payload, CF_WriteQueue_Payload::queue, and CF_WriteQueue_Payload::type.

Here is the call graph for this function:



12.39 apps/cf/fsw/src(cf_codec.c File Reference)

```
#include "cf_app.h"
#include "cf_cfdp_pdu.h"
#include "cf_codec.h"
#include "cf_events.h"
#include <stdint.h>
```

Data Structures

- struct CF_Codec_BitField

Macros

- `#define CF_INIT_FIELD(NBITS, SHIFT)`
 - `#define FGV(SRC, NAME) (CF_FieldGetVal((SRC).octets, (NAME).shift, (NAME).mask))`
 - `#define FSV(DEST, NAME, VAL) (CF_FieldSetVal((DEST).octets, (NAME).shift, (NAME).mask, VAL))`

TypeDefs

- `typedef struct CF_Codec_BitField CF_Codec_BitField_t`

Functions

- static uint8 CF_FieldGetVal (const uint8 *src, uint8 shift, uint8 mask)
 - static void CF_FieldSetVal (uint8 *dest, uint8 shift, uint8 mask, uint8 val)
 - static void CF_Codec_Store_uint8 (CF_CFDP_uint8_t *pdst, uint8 val)
 - static void CF_Codec_Store_uint16 (CF_CFDP_uint16_t *pdst, uint16 val)
 - static void CF_Codec_Store_uint32 (CF_CFDP_uint32_t *pdst, uint32 val)
 - static void CF_Codec_Store_uint64 (CF_CFDP_uint64_t *pdst, uint64 val)
 - static void CF_Codec_Load_uint8 (uint8 *pdst, const CF_CFDP_uint8_t *psrc)
 - static void CF_Codec_Load_uint16 (uint16 *pdst, const CF_CFDP_uint16_t *psrc)
 - static void CF_Codec_Load_uint32 (uint32 *pdst, const CF_CFDP_uint32_t *psrc)

- static void `CF_Codec_Load_uint64` (`uint64` *pdst, const `CF_CFDP_uint64_t` *psrc)
 - bool `CF_CFDP_CodecCheckSize` (`CF_CodecState_t` *state, `size_t` chunksize)

Advances the position by the indicated size, confirming the block will fit into the PDU.
 - void * `CF_CFDP_DoEncodeChunk` (`CF_EncoderState_t` *state, `size_t` chunksize)

Encode a block of data into the PDU.
- const void * `CF_CFDP_DoDecodeChunk` (`CF_DecoderState_t` *state, `size_t` chunksize)

Decode a block of data from the PDU.
- `uint8 CF_CFDP_GetValueEncodedSize` (`uint64` Value)

Gets the minimum number of octets that the given integer may be encoded in.
- void `CF_EncodeIntegerInSize` (`CF_EncoderState_t` *state, `uint64` value, `uint8` encode_size)

Encodes the given integer value in the given number of octets.
- void `CF_CFDP_EncodeHeaderWithoutSize` (`CF_EncoderState_t` *state, `CF_Logical_PduHeader_t` *plh)

Encodes a CFDP PDU base header block, bypassing the size field.
- void `CF_CFDP_EncodeHeaderFinalSize` (`CF_EncoderState_t` *state, `CF_Logical_PduHeader_t` *plh)

Updates an already-encoded PDU base header block with the final PDU size.
- void `CF_CFDP_EncodeFileDirectiveHeader` (`CF_EncoderState_t` *state, `CF_Logical_PduFileDirectiveHeader_t` *pfdir)

Encodes a CFDP file directive header block.
- void `CF_CFDP_EncodeLV` (`CF_EncoderState_t` *state, `CF_Logical_Lv_t` *pllv)

Encodes a single CFDP Length+Value (LV) pair.
- void `CF_CFDP_EncodeTLV` (`CF_EncoderState_t` *state, `CF_Logical_Tlv_t` *pltlv)

Encodes a single CFDP Type+Length+Value (TLV) tuple.
- void `CF_CFDP_EncodeSegmentRequest` (`CF_EncoderState_t` *state, `CF_Logical_SegmentRequest_t` *plseg)

Encodes a single CFDP Segment Request block.
- void `CF_CFDP_EncodeAllTlv` (`CF_EncoderState_t` *state, `CF_Logical_TlvList_t` *pltlv)

Encodes a list of CFDP Type+Length+Value tuples.
- void `CF_CFDP_EncodeAllSegments` (`CF_EncoderState_t` *state, `CF_Logical_SegmentList_t` *plseg)

Encodes a list of CFDP Segment Request blocks.
- void `CF_CFDP_EncodeMd` (`CF_EncoderState_t` *state, `CF_Logical_PduMd_t` *plmd)

Encodes a CFDP Metadata (MD) header block.
- void `CF_CFDP_EncodeFileDataHeader` (`CF_EncoderState_t` *state, `bool` with_meta, `CF_Logical_PduFileDataHeader_t` *plfd)

Encodes a CFDP File Data (FD) header block.
- void `CF_CFDP_EncodeEof` (`CF_EncoderState_t` *state, `CF_Logical_PduEof_t` *pleof)

Encodes a CFDP End-of-File (EOF) header block.
- void `CF_CFDP_EncodeFin` (`CF_EncoderState_t` *state, `CF_Logical_PduFin_t` *plfin)

Encodes a CFDP Final (FIN) header block.
- void `CF_CFDP_EncodeAck` (`CF_EncoderState_t` *state, `CF_Logical_PduAck_t` *plack)

Encodes a CFDP Acknowledge (ACK) header block.
- void `CF_CFDP_EncodeNak` (`CF_EncoderState_t` *state, `CF_Logical_PduNak_t` *plnak)

Encodes a CFDP Non-Acknowledge (NAK) header block.
- void `CF_CFDP_EncodeCrc` (`CF_EncoderState_t` *state, `uint32` *plcrc)

Encodes a CFDP CRC/Checksum.
- `uint64 CF_DecodeIntegerInSize` (`CF_DecoderState_t` *state, `uint8` decode_size)

Decodes an integer value from the specified number of octets.
- `CFE_Status_t CF_CFDP_DecodeHeader` (`CF_DecoderState_t` *state, `CF_Logical_PduHeader_t` *plh)

Decodes a CFDP base PDU header.

- void `CF_CFDP_DecodeFileDirectiveHeader (CF_DecoderState_t *state, CF_Logical_PduFileDirectiveHeader_t *pfdir)`
Decodes a CFDP file directive header block.
- void `CF_CFDP_DecodeLV (CF_DecoderState_t *state, CF_Logical_Lv_t *pllv)`
Decodes a single CFDP Length+Value (LV) pair.
- void `CF_CFDP_DecodeTLV (CF_DecoderState_t *state, CF_Logical_Tlv_t *pltlv)`
Decodes a single CFDP Type+Length+Value (TLV) tuple.
- void `CF_CFDP_DecodeSegmentRequest (CF_DecoderState_t *state, CF_Logical_SegmentRequest_t *plseg)`
Decodes a single CFDP Segment Request block.
- void `CF_CFDP_DecodeMd (CF_DecoderState_t *state, CF_Logical_PduMd_t *plmd)`
Decodes a CFDP Metadata (MD) header block.
- void `CF_CFDP_DecodeFileDataHeader (CF_DecoderState_t *state, bool with_meta, CF_Logical_PduFileDataHeader_t *plfd)`
Decodes a CFDP File Data (FD) header block.
- void `CF_CFDP_DecodeCrc (CF_DecoderState_t *state, uint32 *plcrc)`
Decodes a CFDP CRC/Checksum.
- void `CF_CFDP_DecodeEof (CF_DecoderState_t *state, CF_Logical_PduEof_t *pleof)`
Decodes a CFDP End-of-File (EOF) header block.
- void `CF_CFDP_DecodeFin (CF_DecoderState_t *state, CF_Logical_PduFin_t *plfin)`
Decodes a CFDP Final (FIN) header block.
- void `CF_CFDP_DecodeAck (CF_DecoderState_t *state, CF_Logical_PduAck_t *plack)`
Decodes a CFDP Acknowledge (ACK) header block.
- void `CF_CFDP_DecodeNak (CF_DecoderState_t *state, CF_Logical_PduNak_t *plnak)`
Decodes a CFDP Non-Acknowledge (NAK) header block.
- void `CF_CFDP_DecodeAllTlv (CF_DecoderState_t *state, CF_Logical_TlvList_t *pltlv, uint8 limit)`
Decodes a list of CFDP Type+Length+Value tuples.
- void `CF_CFDP_DecodeAllSegments (CF_DecoderState_t *state, CF_Logical_SegmentList_t *plseg, uint8 limit)`
Decodes a list of CFDP Segment Request blocks.

Variables

- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_VERSION = CF_INIT_FIELD(3, 5)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_TYPE = CF_INIT_FIELD(1, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_DIR = CF_INIT_FIELD(1, 3)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_MODE = CF_INIT_FIELD(1, 2)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_CRC = CF_INIT_FIELD(1, 1)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_LARGEFILE = CF_INIT_FIELD(1, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_ENTITY = CF_INIT_FIELD(3, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE = CF_INIT_FIELD(3, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduEof_FLAGS_CC = CF_INIT_FIELD(4, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduFin_FLAGS_CC = CF_INIT_FIELD(4, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduFin_FLAGS_DELIVERY_CODE = CF_INIT_FIELD(1, 2)`
- static const `CF_Codec_BitField_t CF_CFDP_PduFin_FLAGS_FILE_STATUS = CF_INIT_FIELD(2, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduAck_DIR_CODE = CF_INIT_FIELD(4, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduAck_DIR_SUBTYPE_CODE = CF_INIT_FIELD(4, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduAck_CC = CF_INIT_FIELD(4, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduAck_TRANSACTION_STATUS = CF_INIT_FIELD(2, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduMd_CLOSURE_REQUESTED = CF_INIT_FIELD(1, 7)`

- static const CF_Codec_BitField_t CF_CFDP_PduMd_CHECKSUM_TYPE = CF_INIT_FIELD(4, 0)
- static const CF_Codec_BitField_t CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE = CF_INIT_FIELD(2, 6)
- static const CF_Codec_BitField_t CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH = CF_INIT_FIELD(6, 0)

12.39.1 Detailed Description

CFDP protocol data structure encode/decode implementation

12.39.2 Macro Definition Documentation

12.39.2.1 CF_INIT_FIELD #define CF_INIT_FIELD(
 NBITS,
 SHIFT)

Value:

```
{
    .shift = (SHIFT), .mask = ((1 << NBITS) - 1) \ }
```

Definition at line 40 of file cf_codec.c.

12.39.2.2 FGV #define FGV(
 SRC,
 NAME) (CF_FieldGetVal((SRC).octets, (NAME).shift, (NAME).mask))

Definition at line 65 of file cf_codec.c.

12.39.2.3 FSV #define FSV(
 DEST,
 NAME,
 VAL) (CF_FieldSetVal((DEST).octets, (NAME).shift, (NAME).mask, VAL))

Definition at line 66 of file cf_codec.c.

12.39.3 Typedef Documentation

12.39.3.1 CF_Codec_BitField_t typedef struct CF_Codec_BitField CF_Codec_BitField_t

12.39.4 Function Documentation

12.39.4.1 CF_CFDP_CodecCheckSize() bool CF_CFDP_CodecCheckSize (
 CF_CodecState_t * state,
 size_t chunksize)

Advances the position by the indicated size, confirming the block will fit into the PDU.

On encode, this confirms there is enough available space to hold a block of the indicated size. On decode, this confirms that decoding the indicated number of bytes will not read beyond the end of data.

If true, then the current position/offset is advanced by the indicated number of bytes. If false, then the error flag is set, so that future calls to CF_CFDP_CodecIsOK will also return false.

Note

The error flag is sticky, meaning that if any encode/decode operation fails, all future encode/decode requests on the same state will also fail. Each encode/decode step must check the flag, and skip the operation if it is false. Reporting the error can be deferred to the final stage, and only done once.

Parameters

<i>state</i>	Encoder/Decoder common state
<i>chunksize</i>	Size of next block to encode/decode

Return values

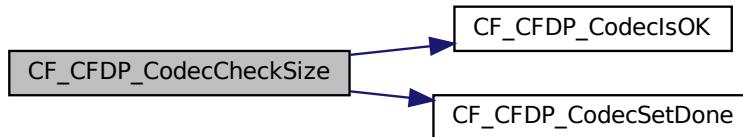
<i>true</i>	If encode/decode is possible, enough space exists
<i>false</i>	If encode/decode is not possible, not enough space or prior error occurred

Definition at line 269 of file cf_codec.c.

References CF_CFDP_CodeclsOK(), CF_CFDP_CodecSetDone(), CF_CodecState::max_size, and CF_CodecState::next_offset.

Referenced by CF_CFDP_DoDecodeChunk(), and CF_CFDP_DoEncodeChunk().

Here is the call graph for this function:



12.39.4.2 CF_CFDP_DecodeAck() void CF_CFDP_DecodeAck (
CF_DecoderState_t * *state*,
CF_Logical_PduAck_t * *plack*)

Decodes a CFDP Acknowledge (ACK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 1043 of file cf_codec.c.

References CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::

::cc, CF_CFDP_PduAck::cc_and_transaction_status, CF_CFDP_PduAck_CC, CF_CFDP_PduAck_DIR_CODE, CF_CFDP_PduAck_DIR_SUBTYPE_CODE, CF_CFDP_PduAck_TRANSACTION_STATUS, CF_DECODE_FIXED_CHUNK, CF_CFDP_PduAck::directive_and_subtype_code, FGV, and CF_Logical_PduAck::txn_status.
Referenced by CF_CFDP_RecvAck().

12.39.4.3 CF_CFDP_DecodeAllSegments() void CF_CFDP_DecodeAllSegments (

```
CF_DecoderState_t * state,
CF_Logical_SegmentList_t * plseg,
uint8 limit )
```

Decodes a list of CFDP Segment Request blocks.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data
<i>limit</i>	Maximum number of Segment Request objects to decode

Definition at line 1119 of file cf_codec.c.

References CF_CFDP_DecodeSegmentRequest(), CF_CODEC_GET_REMAIN, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_PDU_MAX_SEGMENTS, CF_Logical_SegmentList::num_segments, and CF_Logical_SegmentList::segments.

Referenced by CF_CFDP_DecodeNak().

Here is the call graph for this function:



12.39.4.4 CF_CFDP_DecodeAllTlv() void CF_CFDP_DecodeAllTlv (

```
CF_DecoderState_t * state,
CF_Logical_TlvList_t * pltlv,
uint8 limit )
```

Decodes a list of CFDP Type+Length+Value tuples.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

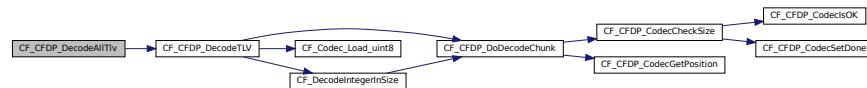
<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data
<i>limit</i>	Maximum number of TLV objects to decode

Definition at line 1084 of file cf_codec.c.

References CF_CFDP_DecodeTLV(), CF_CODEC_GET_REMAIN, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_PDU_MAX_TLV, CF_Logical_TlvList::num_tlv, and CF_Logical_TlvList::tlv.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_DecodeFin().

Here is the call graph for this function:



12.39.4.5 CF_CFDP_DecodeCrc() `void CF_CFDP_DecodeCrc (`

<code>CF_DecoderState_t * state,</code>
<code>uint32 * plcrc)</code>

Decodes a CFDP CRC/Checksum.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<code>state</code>	Decoder state object
<code>plcrc</code>	Pointer to logical CRC value

Definition at line 984 of file cf_codec.c.

References CF_Codec_Load_uint32(), and CF_DECODE_FIXED_CHUNK.

Here is the call graph for this function:



12.39.4.6 CF_CFDP_DecodeEof() `void CF_CFDP_DecodeEof (`

<code>CF_DecoderState_t * state,</code>
<code>CF_Logical_PduEof_t * pleof)</code>

Decodes a CFDP End-of-File (EOF) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<code>state</code>	Decoder state object
--------------------	----------------------

Parameters

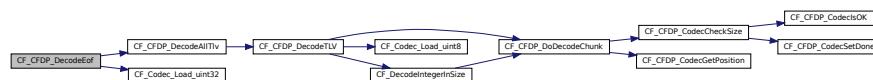
<code>pleof</code>	Pointer to logical PDU EOF header data
--------------------	--

Definition at line 1001 of file cf_codec.c.

References CF_Logical_PduEof::cc, CF_CFDP_PduEof::cc, CF_CFDP_DecodeAllTlv(), CF_CFDP_PduEof_FLAGS←_CC, CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_TLV, CF_Logical_PduEof::crc, CF_CFDP_PduEof::crc, FGV, CF_Logical_PduEof::size, CF_CFDP_PduEof::size, and CF_Logical_PduEof::tlv_list.

Referenced by CF_CFDP_RecvEof().

Here is the call graph for this function:



12.39.4.7 CF_CFDP_DecodeFileDataHeader() `void CF_CFDP_DecodeFileDataHeader (`
`CF_DecoderState_t * state,`
`bool with_meta,`
`CF_Logical_PduFileDataHeader_t * plfd)`

Decodes a CFDP File Data (FD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

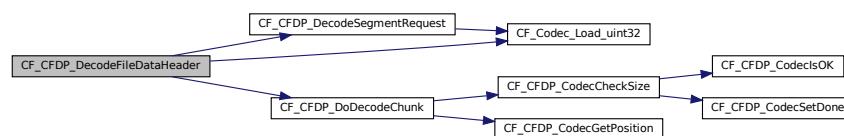
<code>state</code>	Decoder state object
<code>with_meta</code>	Whether to include optional continuation and segment request fields (always false currently)
<code>plfd</code>	Pointer to logical PDU file header data

Definition at line 922 of file cf_codec.c.

References CF_CFDP_DecodeSegmentRequest(), CF_CFDP_DoDecodeChunk(), CF_CFDP_PduFileData_RECO←_RD_CONTINUATION_STATE, CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH, CF_CODEC_GET_R←_EMAIN, CF_CODEC_IS_OK, CF_Codec_Load_uint32(), CF_CODEC_SET_DONE, CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_SEGMENTS, CF_Logical_PduFileDataHeader::continuation_state, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, FGV, CF_Logical_SegmentList::num_segments, CF_Logical_PduFileDataHeader::offset, CF_CFDP_PduFileDataHeader::offset, CF_Logical_PduFileDataHeader::segment_list, and CF_Logical_SegmentList::segments.

Referenced by CF_CFDP_RecvFd().

Here is the call graph for this function:



```
12.39.4.8 CF_CFDP_DecodeFileDirectiveHeader() void CF_CFDP_DecodeFileDirectiveHeader (
    CF_DecoderState_t * state,
    CF_Logical_PduFileDirectiveHeader_t * pfdir )
```

Decodes a CFDP file directive header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 812 of file cf_codec.c.

References CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_Logical_PduFileDirectiveHeader::directive_code, and CF_CFDP_PduFileDirectiveHeader::directive_code.

Referenced by CF_CFDP_RecvPh().

Here is the call graph for this function:



```
12.39.4.9 CF_CFDP_DecodeFin() void CF_CFDP_DecodeFin (
    CF_DecoderState_t * state,
    CF_Logical_PduFin_t * plfin )
```

Decodes a CFDP Final (FIN) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>plfin</i>	Pointer to logical PDU FIN header data

Definition at line 1022 of file cf_codec.c.

References CF_Logical_PduFin::cc, CF_CFDP_DecodeAllTlv(), CF_CFDP_PduFin_FLAGS_CC, CF_CFDP_PduFin_FLAGS_DELIVERY_CODE, CF_CFDP_PduFin_FLAGS_FILE_STATUS, CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_TLV, CF_Logical_PduFin::delivery_code, FGV, CF_Logical_PduFin::file_status, CF_CFDP_PduFin::flags, and CF_Logical_PduFin::tlv_list.

Referenced by CF_CFDP_RecvFin().

Here is the call graph for this function:



12.39.4.10 CF_CFDP_DecodeHeader()

```
CFE_Status_t CF_CFDP_DecodeHeader (
    CF_DecoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Decodes a CFDP base PDU header.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure.

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Note

On decode the entire base header is decoded in a single call, the size will be decoded like any other field.

Parameters

<i>state</i>	Decoder state object
<i>plh</i>	Pointer to logical PDU base header data

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>CF_ERROR</i>	on error.

Definition at line 766 of file cf_codec.c.

References CF_CFDP_PduHeader_FLAGS_CRC, CF_CFDP_PduHeader_FLAGS_DIR, CF_CFDP_PduHeader_FLAGS_MODE, CF_CFDP_PduHeader_FLAGS_TYPE, CF_CFDP_PduHeader_FLAGS_VERSION, CF_CFDP_PduHeader_LENGTHS_ENTITY, CF_CFDP_PduHeader_LENGTHS_TRACTIONSEQUENCE, CF_CODEC_GET_POSITION, CF_Codec_Load_uint16(), CF_DECODE_FIXED_CHUNK, CF_DecodeIntegerInSize(), CF_ERROR, CFE_SUCCESS, CF_Logical_PduHeader::crc_flag, CF_Logical_PduHeader::data_encoded_length, CF_Logical_PduHeader::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduHeader::eid_length, CF_CFDP_PduHeader::eid_tsn_lengths, FGV, CF_CFDP_PduHeader::flags, CF_Logical_PduHeader::header_encoded_length, CF_Logical_PduHeader::large_flag, CF_CFDP_PduHeader::length, CF_Logical_PduHeader::pdu_type, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_RecvPh().

Here is the call graph for this function:



```
12.39.4.11 CF_CFDP_DecodeLV() void CF_CFDP_DecodeLV (
    CF_DecoderState_t * state,
    CF_Logical_Lv_t * pllv )
```

Decodes a single CFDP Length+Value (LV) pair.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

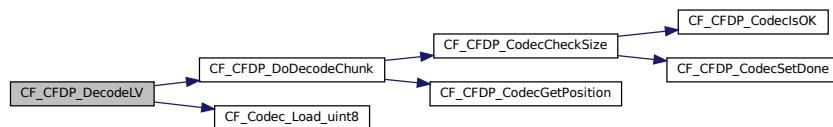
<i>state</i>	Decoder state object
<i>pllv</i>	Pointer to single logical PDU LV data

Definition at line 832 of file cf_codec.c.

References CF_CFDP_DoDecodeChunk(), CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_Logical_Lv::data_ptr, CF_Logical_Lv::length, and CF_CFDP_Lv::length.

Referenced by CF_CFDP_DecodeMd().

Here is the call graph for this function:



```
12.39.4.12 CF_CFDP_DecodeMd() void CF_CFDP_DecodeMd (
    CF_DecoderState_t * state,
    CF_Logical_PduMd_t * plmd )
```

Decodes a CFDP Metadata (MD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

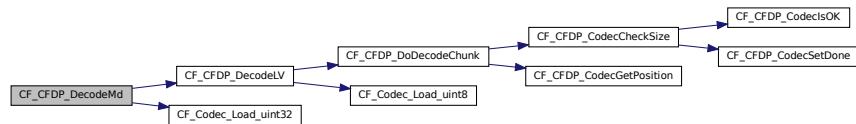
<i>state</i>	Decoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 899 of file cf_codec.c.

References CF_CFDP_DecodeLV(), CF_CFDP_PduMd_CHECKSUM_TYPE, CF_CFDP_PduMd_CLOSURE_REQUESTED, CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_Logical_PduMd::checksum_type, CF_Logical_PduMd::close_req, CF_Logical_PduMd::dest_filename, FGV, CF_CFDP_PduMd::segmentation_control, CF_Logical_PduMd::size, CF_CFDP_PduMd::size, and CF_Logical_PduMd::source_filename.

Referenced by CF_CFDP_RecvMd().

Here is the call graph for this function:



12.39.4.13 CF_CFDP_DecodeNak() `void CF_CFDP_DecodeNak (`
`CF_DecoderState_t * state,`
`CF_Logical_PduNak_t * plnak)`

Decodes a CFDP Non-Acknowledge (NAK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

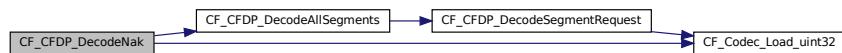
<code>state</code>	Decoder state object
<code>plnak</code>	Pointer to logical PDU NAK header data

Definition at line 1064 of file cf_codec.c.

References `CF_CFDP_DecodeAllSegments()`, `CF_Codec_Load_uint32()`, `CF_DECODE_FIXED_CHUNK`, `CF_PDU_MAX_SEGMENTS`, `CF_Logical_PduNak::scope_end`, `CF_CFDP_PduNak::scope_end`, `CF_Logical_PduNak::scope_start`, `CF_CFDP_PduNak::scope_start`, and `CF_Logical_PduNak::segment_list`.

Referenced by `CF_CFDP_RecvNak()`.

Here is the call graph for this function:



12.39.4.14 CF_CFDP_DecodeSegmentRequest() `void CF_CFDP_DecodeSegmentRequest (`
`CF_DecoderState_t * state,`
`CF_Logical_SegmentRequest_t * plseg)`

Decodes a single CFDP Segment Request block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<code>state</code>	Decoder state object
<code>plseg</code>	Pointer to single logical PDU segment request header data

Definition at line 881 of file cf_codec.c.

References CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_Logical_SegmentRequest::offset_end, CF_CFDP_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, and CF_CFDP_SegmentRequest::offset_start.

Referenced by CF_CFDP_DecodeAllSegments(), and CF_CFDP_DecodeFileDataHeader().

Here is the call graph for this function:



12.39.4.15 CF_CFDP_DecodeTLV()

```
void CF_CFDP_DecodeTLV (
    CF_DecoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Decodes a single CFDP Type+Length+Value (TLV) tuple.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV data

Definition at line 850 of file cf_codec.c.

References CF_CFDP_DoDecodeChunk(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_DecodeIntegerInSize(), CF_Logical_Tlv::data, CF_Logical_TlvData::data_ptr, CF_Logical_TlvData::eid, CF_Logical_Tlv::length, CF_CFDP_tlv::length, CF_Logical_Tlv::type, and CF_CFDP_tlv::type.

Referenced by CF_CFDP_DecodeAllTlv().

Here is the call graph for this function:



12.39.4.16 CF_CFDP_DoDecodeChunk()

```
const void* CF_CFDP_DoDecodeChunk (
    CF_DecoderState_t * state,
    size_t chunksize )
```

Decode a block of data from the PDU.

Deducts space for a block of the given size from the current PDU

Parameters

<i>state</i>	Decoder state object
<i>chunksize</i>	Size of block to decode

Returns

Pointer to block, if successful

Return values

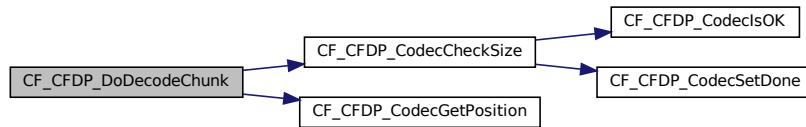
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 309 of file cf_codec.c.

References CF_DecoderState::base, CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetPosition(), and CF_DecoderState::codec_state.

Referenced by CF_CFDP_DecodeFileDataHeader(), CF_CFDP_DecodeLV(), CF_CFDP_DecodeTLV(), and CF_DecodeIntegerInSize().

Here is the call graph for this function:

**12.39.4.17 CF_CFDP_DoEncodeChunk()** void* CF_CFDP_DoEncodeChunk (

```
CF_EncoderState_t * state,
size_t chunksize )
```

Encode a block of data into the PDU.

Adds/Reserves space for a block of the given size in the current PDU

Parameters

<i>state</i>	Encoder state object
<i>chunksize</i>	Size of block to encode

Returns

Pointer to block, if successful

Return values

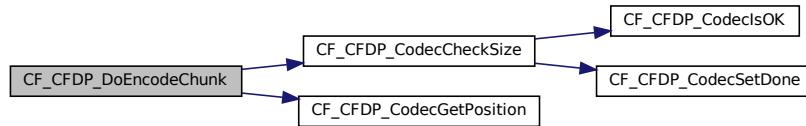
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 291 of file cf_codec.c.

References CF_EncoderState::base, CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetPosition(), and CF_EncoderState::codec_state.

Referenced by CF_CFDP_EncodeLV(), CF_CFDP_EncodeTLV(), CF_CFDP_S_SendFileData(), and CF_EncodeIntegerInSize().

Here is the call graph for this function:



12.39.4.18 CF_CFDP_EncodeAck() `void CF_CFDP_EncodeAck (`
 `CF_EncoderState_t * state,`
 `CF_Logical_PduAck_t * plack)`

Encodes a CFDP Acknowledge (ACK) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<code>state</code>	Encoder state object
<code>plack</code>	Pointer to logical PDU ACK header data

Definition at line 678 of file cf_codec.c.

References CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::cc, CF_CFDP_PduAck::cc_and_transaction_status, CF_CFDP_PduAck_CC, CF_CFDP_PduAck_DIR_CODE, CF_CFDP_PduAck_DIR_SUBTYPE_CODE, CF_CFDP_PduAck_TRANSACTION_STATUS, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_CFDP_PduAck::directive_and_subtype_code, FSV, and CF_Logical_PduAck::tx_status.

Referenced by CF_CFDP_SendAck().

Here is the call graph for this function:



12.39.4.19 CF_CFDP_EncodeAllSegments() `void CF_CFDP_EncodeAllSegments (`

```
CF_EncoderState_t * state,
CF_Logical_SegmentList_t * plseg )
```

Encodes a list of CFDP Segment Request blocks.

This invokes [CF_CFDP_EncodeSegmentRequest\(\)](#) for all segments in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

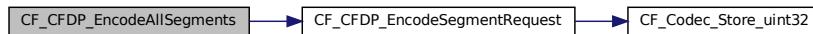
<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data

Definition at line 557 of file cf_codec.c.

References [CF_CFDP_EncodeSegmentRequest\(\)](#), [CF_CODEC_IS_OK](#), [CF_Logical_SegmentList::num_segments](#), and [CF_Logical_SegmentList::segments](#).

Referenced by [CF_CFDP_EncodeFileDataHeader\(\)](#), and [CF_CFDP_EncodeNak\(\)](#).

Here is the call graph for this function:



12.39.4.20 CF_CFDP_EncodeAllTlv() void CF_CFDP_EncodeAllTlv (

```
CF_EncoderState_t * state,
CF_Logical_TlvList_t * pltlv )
```

Encodes a list of CFDP Type+Length+Value tuples.

This invokes [CF_CFDP_EncodeTLV\(\)](#) for all TLV values in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

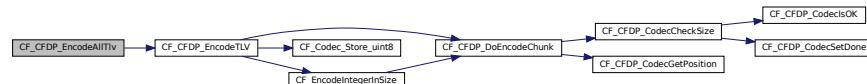
<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data

Definition at line 541 of file cf_codec.c.

References [CF_CFDP_EncodeTLV\(\)](#), [CF_CODEC_IS_OK](#), [CF_Logical_TlvList::num_tlv](#), and [CF_Logical_TlvList::tlv](#).

Referenced by [CF_CFDP_EncodeEof\(\)](#), and [CF_CFDP_EncodeFin\(\)](#).

Here is the call graph for this function:



```
12.39.4.21 CF_CFDP_EncodeCrc() void CF_CFDP_EncodeCrc (
    CF_EncoderState_t * state,
    uint32 * plcrc )
```

Encodes a CFDP CRC/Checksum.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<i>state</i>	Encoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 721 of file cf_codec.c.

References CF_Codec_Store_uint32(), and CF_ENCODE_FIXED_CHUNK.

Here is the call graph for this function:



```
12.39.4.22 CF_CFDP_EncodeEof() void CF_CFDP_EncodeEof (
    CF_EncoderState_t * state,
    CF_Logical_PduEof_t * pleof )
```

Encodes a CFDP End-of-File (EOF) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any TLV values which are indicated in the logical data structure

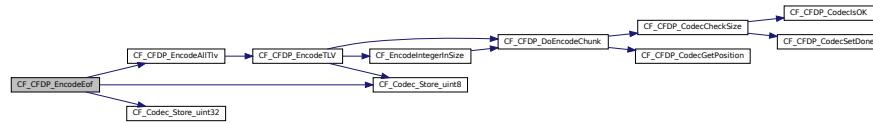
Parameters

<i>state</i>	Encoder state object
<i>pleof</i>	Pointer to logical PDU EOF header data

Definition at line 634 of file cf_codec.c.

References CF_Logical_PduEof::cc, CF_CFDP_PduEof::cc, CF_CFDP_EncodeAllTlv(), CF_CFDP_PduEof_FLAGS←_CC, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduEof::crc, CF_CFDP_PduEof::crc, FSV, CF_Logical_PduEof::size, CF_CFDP_PduEof::size, and CF_Logical_PduEof::tlv_list.
Referenced by CF_CFDP_SendEof().

Here is the call graph for this function:



12.39.4.23 CF_CFDP_EncodeFileDataHeader()

```
void CF_CFDP_EncodeFileDataHeader (
    CF_EncoderState_t * state,
    bool with_meta,
    CF_Logical_PduFileDataHeader_t * plfd )
```

Encodes a CFDP File Data (FD) header block.

This only encodes the FD header fields, specifically the data offset (required) and any metadata fields, if indicated. This does *not* encode any actual file data.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

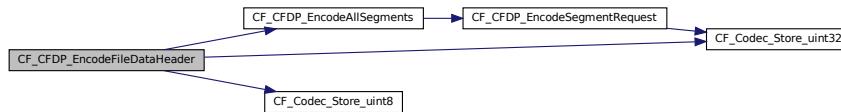
<i>state</i>	Encoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>plfd</i>	Pointer to logical PDU file header data

Definition at line 597 of file cf_codec.c.

References CF_CFDP_EncodeAllSegments(), CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE, CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_E_NCODE_FIXED_CHUNK, CF_Logical_PduFileDataHeader::continuation_state, FSV, CF_Logical_SegmentList::num_segments, CF_Logical_PduFileDataHeader::offset, CF_CFDP_PduFileDataHeader::offset, and CF_Logical_PduFileDataHeader::segment_list.

Referenced by CF_CFDP_S_SendFileData().

Here is the call graph for this function:



12.39.4.24 CF_CFDP_EncodeFileDirectiveHeader()

```
void CF_CFDP_EncodeFileDirectiveHeader (
    CF_EncoderState_t * state,
    CF_Logical_PduFileDirectiveHeader_t * pfdir )
```

Encodes a CFDP file directive header block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<i>state</i>	Encoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 437 of file cf_codec.c.

References CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduFileDirectiveHeader::directive_code, and CF_CFDP_PduFileDirectiveHeader::directive_code.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.39.4.25 CF_CFDP_EncodeFin() void CF_CFDP_EncodeFin (

```

    CF_EncoderState_t * state,
    CF_Logical_PduFin_t * plfin )

```

Encodes a CFDP Final (FIN) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any TLV values which are indicated in the logical data structure

Parameters

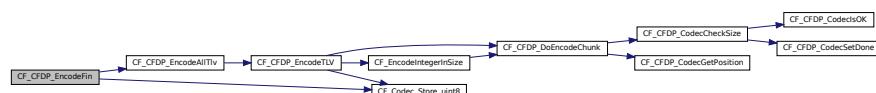
<i>state</i>	Encoder state object
<i>plfin</i>	Pointer to logical PDU FIN header data

Definition at line 656 of file cf_codec.c.

References CF_Logical_PduFin::cc, CF_CFDP_EncodeAllTlv(), CF_CFDP_PduFin_FLAGS_CC, CF_CFDP_PduFin::FLAGS_DELIVERY_CODE, CF_CFDP_PduFin_FLAGS_FILE_STATUS, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduFin::delivery_code, CF_Logical_PduFin::file_status, CF_CFDP_PduFin::flags, FSV, and CF_Logical_PduFin::tlv_list.

Referenced by CF_CFDP_SendFin().

Here is the call graph for this function:



```
12.39.4.26 CF_CFDP_EncodeHeaderFinalSize() void CF_CFDP_EncodeHeaderFinalSize (
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Updates an already-encoded PDU base header block with the final PDU size.

This function encodes the "data_encoded_length" field from the logical PDU structure into the encoded header block. The PDU will also be closed (set done) to indicate that no more data should be added.

Note

Unlike other encode operations, this function does not add any new blocks to the PDU. It only updates the already-encoded block at the beginning of the PDU, which must have been done by a prior call to [CF_CFDP_EncodeHeaderWithoutSize\(\)](#).

See also

[CF_CFDP_EncodeHeaderWithoutSize\(\)](#) for initially encoding the PDU header block

Parameters

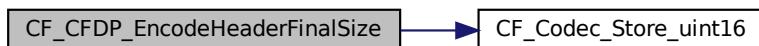
<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 407 of file cf_codec.c.

References CF_EncoderState::base, CF_CODEC_GET_POSITION, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_Codec_Store_uint16(), CF_Logical_PduHeader::data_encoded_length, and CF_CFDP_PduHeader::length.

Referenced by CF_CFDP_SetPduLength().

Here is the call graph for this function:



```
12.39.4.27 CF_CFDP_EncodeHeaderWithoutSize() void CF_CFDP_EncodeHeaderWithoutSize (
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Encodes a CFDP PDU base header block, bypassing the size field.

On transmit side, the common/base header must be encoded in two parts, to deal with the "total_size" field. The initial encoding of the basic fields is done as soon as it is known that a PDU of this type needs to be sent, but the total size may not be yet known, as it depends on the remainder of encoding and any additional data that might get added to the variable length sections.

This function encodes all base header fields *except* total length. There is a separate function later to update the total_length to the correct value once the remainder of encoding is done. Luckily, the total_length is in the first fixed position binary blob so it is easy to update later.

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

See also

[CF_CFDP_EncodeHeaderFinalSize\(\)](#) for updating the length field once it is known

Parameters

<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 371 of file cf_codec.c.

References CF_CFDP_PduHeader_FLAGS_DIR, CF_CFDP_PduHeader_FLAGS_MODE, CF_CFDP_PduHeader→_FLAGS_TYPE, CF_CFDP_PduHeader_FLAGS_VERSION, CF_CFDP_PduHeader_LENGTHS_ENTITY, CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE, CF_CODEC_GET_POSITION, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_EncodeIntegerInSize(), CF_Logical_PduHeader::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduHeader::eid_length, CF_CFDP_PduHeader::eid_tsn_lengths, CF_CFDP_PduHeader::flags, FSV, CF_Logical_PduHeader::header_encoded_length, CF_Logical_PduHeader::pdu_type, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.39.4.28 CF_CFDP_EncodeLV() void CF_CFDP_EncodeLV (
CF_EncoderState_t * *state*,
CF_Logical_Lv_t * *pllv*)

Encodes a single CFDP Length+Value (LV) pair.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

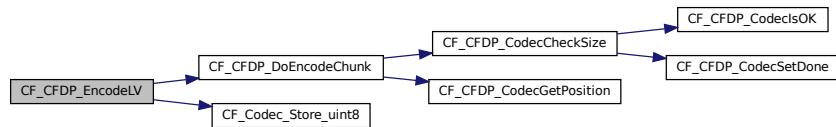
<i>state</i>	Encoder state object
<i>pllv</i>	Pointer to logical PDU LV header data

Definition at line 455 of file cf_codec.c.

References CF_CFDP_DoEncodeChunk(), CF_CODEC_SET_DONE, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_Lv::data_ptr, CF_Logical_Lv::length, and CF_CFDP_Lv::length.

Referenced by CF_CFDP_EncodeMd().

Here is the call graph for this function:



12.39.4.29 CF_CFDP_EncodeMd() `void CF_CFDP_EncodeMd (`
`CF_EncoderState_t * state,`
`CF_Logical_PduMd_t * plmd)`

Encodes a CFDP Metadata (MD) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes the LV pairs for source and destination file names, which are logically part of the overall MD block.

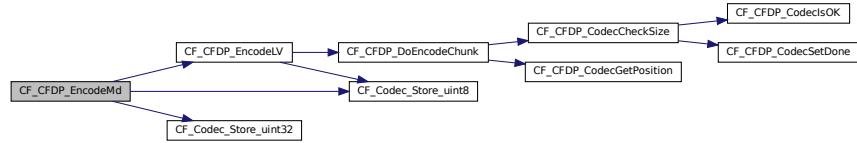
Parameters

<code>state</code>	Encoder state object
<code>plmd</code>	Pointer to logical PDU metadata header data

Definition at line 573 of file cf_codec.c.

References CF_CFDP_EncodeLV(), CF_CFDP_PduMd_CHECKSUM_TYPE, CF_CFDP_PduMd_CLOSURE_REQ, UESTED, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduMd::checksum_type, CF_Logical_PduMd::close_req, CF_Logical_PduMd::dest_filename, FSV, CF_CFDP_PduMd::segmentation_control, CF_Logical_PduMd::size, CF_CFDP_PduMd::size, and CF_Logical_PduMd::source_filename. Referenced by CF_CFDP_SendMd().

Here is the call graph for this function:



12.39.4.30 CF_CFDP_EncodeNak() `void CF_CFDP_EncodeNak (`
`CF_EncoderState_t * state,`
`CF_Logical_PduNak_t * plnak)`

Encodes a CFDP Non-Acknowledge (NAK) header block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any Segment Request values which are indicated in the logical data structure

Parameters

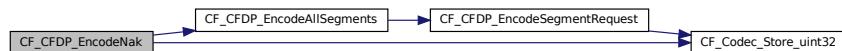
<i>state</i>	Encoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 701 of file cf_codec.c.

References CF_CFDP_EncodeAllSegments(), CF_Codec_Store_uint32(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduNak::scope_end, CF_CFDP_PduNak::scope_end, CF_Logical_PduNak::scope_start, CF_CFDP_PduNak::scope_start, and CF_Logical_PduNak::segment_list.

Referenced by CF_CFDP_SendNak().

Here is the call graph for this function:



12.39.4.31 CF_CFDP_EncodeSegmentRequest()

```
void CF_CFDP_EncodeSegmentRequest (
    CF_EncoderState_t * state,
    CF_Logical_SegmentRequest_t * plseg )
```

Encodes a single CFDP Segment Request block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to single logical PDU segment request header data

Definition at line 523 of file cf_codec.c.

References CF_Codec_Store_uint32(), CF_ENCODE_FIXED_CHUNK, CF_Logical_SegmentRequest::offset_end, CF_CFDP_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, and CF_CFDP_SegmentRequest::offset_start.

Referenced by CF_CFDP_EncodeAllSegments().

Here is the call graph for this function:



```
12.39.4.32 CF_CFDP_EncodeTLV() void CF_CFDP_EncodeTLV (
    CF_EncoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Encodes a single CFDP Type+Length+Value (TLV) tuple.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

Only the CF_CFDP_TLV_TYPE_ENTITY_ID TLV type is currently supported by this function, but other TLV types may be added in future versions as needed.

Parameters

<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV header data

Definition at line 485 of file cf_codec.c.

References CF_CFDP_DoEncodeChunk(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_CODEC_SET_DONE, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_EncodeIntegerInSize(), CF_Logical_Tlv::data, CF_Logical_TlvData::data_ptr, CF_Logical_TlvData::eid, CF_Logical_Tlv::length, CF_CFDP_tlv::length, CF_CFDP_tlv::type, and CF_Logical_Tlv::type.

Referenced by CF_CFDP_EncodeAllTlv().

Here is the call graph for this function:



```
12.39.4.33 CF_CFDP_GetValueEncodedSize() uint8 CF_CFDP_GetValueEncodedSize (
    uint64 Value )
```

Gets the minimum number of octets that the given integer may be encoded in.

Based on the integer value, this computes the minimum number of bytes that must be allocated to that integer within a CFDP PDU. This is typically used for entity IDs and sequence numbers, where CFDP does not specify a specific size for these items. They may be encoded between 1 and 8 bytes, depending on the actual value is.

Parameters

<i>Value</i>	Integer value that needs to be encoded
--------------	--

Returns

Minimum number of bytes that the value requires (between 1 and 8, inclusive)

Definition at line 327 of file cf_codec.c.

Referenced by CF_CFDP_AppendTlv(), and CF_CFDP_ConstructPduHeader().

12.39.4.34 CF_Codec_Load_uint16() static void CF_Codec_Load_uint16 (

```
    uint16 * pdst,
    const CF_CFDP_uint16_t * psrc ) [inline], [static]
```

Definition at line 204 of file cf_codec.c.

References CF_CFDP_uint16_t::octets.

Referenced by CF_CFDP_DecodeHeader().

12.39.4.35 CF_Codec_Load_uint32() static void CF_Codec_Load_uint32 (

```
    uint32 * pdst,
    const CF_CFDP_uint32_t * psrc ) [inline], [static]
```

Definition at line 220 of file cf_codec.c.

References CF_CFDP_uint32_t::octets.

Referenced by CF_CFDP_DecodeCrc(), CF_CFDP_DecodeEof(), CF_CFDP_DecodeFileDataHeader(), CF_CFDP_DecodeMd(), CF_CFDP_DecodeNak(), and CF_CFDP_DecodeSegmentRequest().

12.39.4.36 CF_Codec_Load_uint64() static void CF_Codec_Load_uint64 (

```
    uint64 * pdst,
    const CF_CFDP_uint64_t * psrc ) [inline], [static]
```

Definition at line 240 of file cf_codec.c.

References CF_CFDP_uint64_t::octets.

12.39.4.37 CF_Codec_Load_uint8() static void CF_Codec_Load_uint8 (

```
    uint8 * pdst,
    const CF_CFDP_uint8_t * psrc ) [inline], [static]
```

Definition at line 194 of file cf_codec.c.

References CF_CFDP_uint8_t::octets.

Referenced by CF_CFDP_DecodeFileDirectiveHeader(), CF_CFDP_DecodeLV(), and CF_CFDP_DecodeTLV().

12.39.4.38 CF_Codec_Store_uint16() static void CF_Codec_Store_uint16 (

```
    CF_CFDP_uint16_t * pdst,
    uint16 val ) [inline], [static]
```

Definition at line 142 of file cf_codec.c.

References CF_CFDP_uint16_t::octets.

Referenced by CF_CFDP_EncodeHeaderFinalSize().

12.39.4.39 CF_Codec_Store_uint32() static void CF_Codec_Store_uint32 (

```
    CF_CFDP_uint32_t * pdst,
    uint32 val ) [inline], [static]
```

Definition at line 154 of file cf_codec.c.

References CF_CFDP_uint32_t::octets.

Referenced by CF_CFDP_EncodeCrc(), CF_CFDP_EncodeEof(), CF_CFDP_EncodeFileDataHeader(), CF_CFDP_EncodeMd(), CF_CFDP_EncodeNak(), and CF_CFDP_EncodeSegmentRequest().

```
12.39.4.40 CF_Codec_Store_uint64() static void CF_Codec_Store_uint64 (
    CF_CFDP_uint64_t * pdst,
    uint64 val ) [inline], [static]
```

Definition at line 170 of file cf_codec.c.

References CF_CFDP_uint64_t::octets.

```
12.39.4.41 CF_Codec_Store_uint8() static void CF_Codec_Store_uint8 (
    CF_CFDP_uint8_t * pdst,
    uint8 val ) [inline], [static]
```

Definition at line 132 of file cf_codec.c.

References CF_CFDP_uint8_t::octets.

Referenced by CF_CFDP_EncodeAck(), CF_CFDP_EncodeEof(), CF_CFDP_EncodeFileDataHeader(), CF_CFDP_EncodeFileDirectiveHeader(), CF_CFDP_EncodeFin(), CF_CFDP_EncodeHeaderWithoutSize(), CF_CFDP_EncodeLV(), CF_CFDP_EncodeMd(), and CF_CFDP_EncodeTLV().

```
12.39.4.42 CF_DecodeIntegerInSize() uint64 CF_DecodeIntegerInSize (
    CF_DecoderState_t * state,
    uint8 decode_size )
```

Decodes an integer value from the specified number of octets.

This decodes an integer value in the specified number of octets. The actual number of octets must be determined using another field in the PDU before calling this function.

Warning

This function will decode exactly the given number of octets. If this does not match actual encoded size, the return value will be wrong, and it will likely also throw off the decoding of any fields that follow this one.

See also

[CF_EncodeIntegerInSize\(\)](#) for the inverse operation

Parameters

<i>state</i>	Encoder state object
<i>decode_size</i>	Number of octets that the value is encoded in (between 1 and 8, inclusive)

Returns

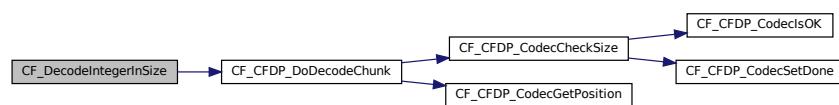
Decoded value

Definition at line 738 of file cf_codec.c.

References CF_CFDP_DoDecodeChunk().

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_DecodeTLV().

Here is the call graph for this function:



```
12.39.4.43 CF_EncodeIntegerInSize() void CF_EncodeIntegerInSize (
    CF_EncoderState_t * state,
    uint64 value,
    uint8 encode_size )
```

Encodes the given integer value in the given number of octets.

This encodes an integer value in the specified number of octets. Use [CF_CFDP_GetValueEncodedSize\(\)](#) to determine the minimum number of octets required for a given value. Using more than the minimum is OK, but will consume extra bytes.

Warning

This function does not error check the encode_size parameter, and will encode the size given, even if it results in an invalid value. Using fewer octets than the minimum reported by [CF_CFDP_GetValueEncodedSize\(\)](#) will likely result in incorrect decoding at the receiver.

See also

[CF_DecodeIntegerInSize\(\)](#) for the inverse operation

Parameters

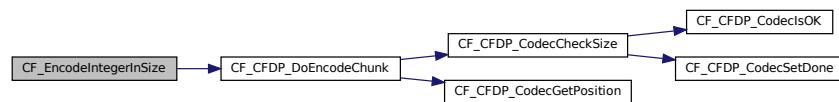
<i>state</i>	Encoder state object
<i>value</i>	Integer value that needs to be encoded
<i>encode_size</i>	Number of octets to encode the value in (between 1 and 8, inclusive)

Definition at line 346 of file cf_codec.c.

References [CF_CFDP_DoEncodeChunk\(\)](#).

Referenced by [CF_CFDP_EncodeHeaderWithoutSize\(\)](#), and [CF_CFDP_EncodeTLV\(\)](#).

Here is the call graph for this function:



```
12.39.4.44 CF_FieldGetVal() static uint8 CF_FieldGetVal (
    const uint8 * src,
    uint8 shift,
    uint8 mask ) [inline], [static]
```

Definition at line 48 of file cf_codec.c.

```
12.39.4.45 CF_FieldSetVal() static void CF_FieldSetVal (
    uint8 * dest,
    uint8 shift,
```

```
    uint8 mask,  
    uint8 val ) [inline], [static]
```

Definition at line 53 of file cf_codec.c.

12.39.5 Variable Documentation

12.39.5.1 CF_CFDP_PduAck_CC const CF_Codec_BitField_t CF_CFDP_PduAck_CC = CF_INIT_FIELD(4, 4)
[static]

Definition at line 102 of file cf_codec.c.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

12.39.5.2 CF_CFDP_PduAck_DIR_CODE const CF_Codec_BitField_t CF_CFDP_PduAck_DIR_CODE = CF_INIT_FIELD(4,
4) [static]

Definition at line 100 of file cf_codec.c.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

12.39.5.3 CF_CFDP_PduAck_DIR_SUBTYPE_CODE const CF_Codec_BitField_t CF_CFDP_PduAck_DIR_SU←
BTYPE_CODE = CF_INIT_FIELD(4, 0) [static]

Definition at line 101 of file cf_codec.c.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

12.39.5.4 CF_CFDP_PduAck_TRANSACTION_STATUS const CF_Codec_BitField_t CF_CFDP_PduAck_TRAN←
SACTION_STATUS = CF_INIT_FIELD(2, 0) [static]

Definition at line 103 of file cf_codec.c.

Referenced by CF_CFDP_DecodeAck(), and CF_CFDP_EncodeAck().

12.39.5.5 CF_CFDP_PduEof_FLAGS_CC const CF_Codec_BitField_t CF_CFDP_PduEof_FLAGS_CC = CF_INIT_FIELD(4,
4) [static]

Definition at line 87 of file cf_codec.c.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_EncodeEof().

12.39.5.6 CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE const CF_Codec_BitField_t CF_CFDP←
_PduFileData_RECORD_CONTINUATION_STATE = CF_INIT_FIELD(2, 6) [static]

Definition at line 117 of file cf_codec.c.

Referenced by CF_CFDP_DecodeFileDataHeader(), and CF_CFDP_EncodeFileDataHeader().

12.39.5.7 CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH const CF_Codec_BitField_t CF_CFDP←
PduFileData_SEGMENT_METADATA_LENGTH = CF_INIT_FIELD(6, 0) [static]

Definition at line 118 of file cf_codec.c.

Referenced by CF_CFDP_DecodeFileDataHeader(), and CF_CFDP_EncodeFileDataHeader().

12.39.5.8 CF_CFDP_PduFin_FLAGS_CC const `CF_Codec_BitField_t` CF_CFDP_PduFin_FLAGS_CC = `CF_INIT_FIELD(4, 4)` [static]

Definition at line 92 of file cf_codec.c.

Referenced by CF_CFDP_DecodeFin(), and CF_CFDP_EncodeFin().

12.39.5.9 CF_CFDP_PduFin_FLAGS_DELIVERY_CODE const `CF_Codec_BitField_t` CF_CFDP_PduFin_FLAGS_DELIVERY_CODE = `CF_INIT_FIELD(1, 2)` [static]

Definition at line 93 of file cf_codec.c.

Referenced by CF_CFDP_DecodeFin(), and CF_CFDP_EncodeFin().

12.39.5.10 CF_CFDP_PduFin_FLAGS_FILE_STATUS const `CF_Codec_BitField_t` CF_CFDP_PduFin_FLAGS_FILE_STATUS = `CF_INIT_FIELD(2, 0)` [static]

Definition at line 94 of file cf_codec.c.

Referenced by CF_CFDP_DecodeFin(), and CF_CFDP_EncodeFin().

12.39.5.11 CF_CFDP_PduHeader_FLAGS_CRC const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_CRC = `CF_INIT_FIELD(1, 1)` [static]

Definition at line 75 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader().

12.39.5.12 CF_CFDP_PduHeader_FLAGS_DIR const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_DIR = `CF_INIT_FIELD(1, 3)` [static]

Definition at line 73 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.13 CF_CFDP_PduHeader_FLAGS_LARGEFILE const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_LARGEFILE = `CF_INIT_FIELD(1, 0)` [static]

Definition at line 76 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader().

12.39.5.14 CF_CFDP_PduHeader_FLAGS_MODE const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_MODE = `CF_INIT_FIELD(1, 2)` [static]

Definition at line 74 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.15 CF_CFDP_PduHeader_FLAGS_TYPE const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_TYPE = `CF_INIT_FIELD(1, 4)` [static]

Definition at line 72 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.16 CF_CFDP_PduHeader_FLAGS_VERSION const `CF_Codec_BitField_t` CF_CFDP_PduHeader_FLAGS_VERSION = `CF_INIT_FIELD(3, 5)` [static]

Definition at line 71 of file cf_codec.c.

Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.17 CF_CFDP_PduHeader_LENGTHS_ENTITY const CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_ENTITY = CF_INIT_FIELD(3, 4) [static]
 Definition at line 81 of file cf_codec.c.
 Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.18 CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE const CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE = CF_INIT_FIELD(3, 0) [static]
 Definition at line 82 of file cf_codec.c.
 Referenced by CF_CFDP_DecodeHeader(), and CF_CFDP_EncodeHeaderWithoutSize().

12.39.5.19 CF_CFDP_PduMd_CHECKSUM_TYPE const CF_Codec_BitField_t CF_CFDP_PduMd_CHECKSUM_TYPE = CF_INIT_FIELD(4, 0) [static]
 Definition at line 110 of file cf_codec.c.
 Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

12.39.5.20 CF_CFDP_PduMd_CLOSURE_REQUESTED const CF_Codec_BitField_t CF_CFDP_PduMd_CLOSURE_REQUESTED = CF_INIT_FIELD(1, 7) [static]
 Definition at line 109 of file cf_codec.c.
 Referenced by CF_CFDP_DecodeMd(), and CF_CFDP_EncodeMd().

12.40 apps/cf/fsw/src(cf_codec.h File Reference

```
#include "cfe.h"
#include "cf_cfdp_pdu.h"
#include "cf_logical_pdu.h"
```

Data Structures

- struct **CF_CodecState**
Tracks the current state of an encode or decode operation.
- struct **CF_EncoderState**
Current state of an encode operation.
- struct **CF_DecoderState**
Current state of a decode operation.

Macros

- #define **CF_ENCODE_FIXED_CHUNK**(state, type) ((type *)CF_CFDP_DoEncodeChunk(state, sizeof(type)))
Macro to encode a block of a given CFDP type into a PDU.
- #define **CF_DECODE_FIXED_CHUNK**(state, type) ((const type *)CF_CFDP_DoDecodeChunk(state, sizeof(type)))
Macro to decode a block of a given CFDP type into a PDU.
- #define **CF_CODEC_IS_OK**(s) (CF_CFDP_CodecIsOK(&((s)->codec_state)))
Macro wrapper around CF_CFDP_CodecIsOK()
- #define **CF_CODEC_SET_DONE**(s) (CF_CFDP_CodecSetDone(&((s)->codec_state)))

```
Macro wrapper around CF_CFDP_CodecSetDone()
• #define CF_CODEC_GET_POSITION(s) (CF_CFDP_CodecGetPosition(&((s)->codec_state)))

Macro wrapper around CF_CFDP_CodecGetPosition()
• #define CF_CODEC_GET_REMAIN(s) (CF_CFDP_CodecGetRemain(&((s)->codec_state)))

Macro wrapper around CF_CFDP_CodecGetRemain()
• #define CF_CODEC_GET_SIZE(s) (CF_CFDP_CodecGetSize(&((s)->codec_state)))

Macro wrapper around CF_CFDP_CodecGetSize()
```

Typedefs

- **typedef struct CF_CodecState CF_CodecState_t**
Tracks the current state of an encode or decode operation.
- **typedef struct CF_EncoderState CF_EncoderState_t**
Current state of an encode operation.
- **typedef struct CF_DecoderState CF_DecoderState_t**
Current state of a decode operation.

Functions

- **static bool CF_CFDP_CodeclsOK (const CF_CodecState_t *state)**
Checks if the codec is currently valid or not.
- **static void CF_CFDP_CodecSetDone (CF_CodecState_t *state)**
Sets a codec to the "done" state.
- **static size_t CF_CFDP_CodecGetPosition (const CF_CodecState_t *state)**
Obtains the current position/offset within the PDU.
- **static size_t CF_CFDP_CodecGetSize (const CF_CodecState_t *state)**
Obtains the maximum size of the PDU being encoded/decoded.
- **static size_t CF_CFDP_CodecGetRemain (const CF_CodecState_t *state)**
Obtains the remaining size of the PDU being encoded/decoded.
- **static void CF_CFDP_CodecReset (CF_CodecState_t *state, size_t max_size)**
Resets a codec state.
- **bool CF_CFDP_CodecCheckSize (CF_CodecState_t *state, size_t chunksize)**
Advances the position by the indicated size, confirming the block will fit into the PDU.
- **void * CF_CFDP_DoEncodeChunk (CF_EncoderState_t *state, size_t chunksize)**
Encode a block of data into the PDU.
- **const void * CF_CFDP_DoDecodeChunk (CF_DecoderState_t *state, size_t chunksize)**
Decode a block of data from the PDU.
- **uint8 CF_CFDP_GetValueEncodedSize (uint64 Value)**
Gets the minimum number of octets that the given integer may be encoded in.
- **void CF_EncodeIntegerInSize (CF_EncoderState_t *state, uint64 value, uint8 encode_size)**
Encodes the given integer value in the given number of octets.
- **uint64 CF_DecodeIntegerInSize (CF_DecoderState_t *state, uint8 decode_size)**
Decodes an integer value from the specified number of octets.
- **void CF_CFDP_EncodeHeaderWithoutSize (CF_EncoderState_t *state, CF_Logical_PduHeader_t *plh)**
Encodes a CFDP PDU base header block, bypassing the size field.
- **void CF_CFDP_EncodeHeaderFinalSize (CF_EncoderState_t *state, CF_Logical_PduHeader_t *plh)**
Updates an already-encoded PDU base header block with the final PDU size.

- void `CF_CFDP_EncodeFileDirectiveHeader` (`CF_EncoderState_t` *state, `CF_Logical_PduFileDirectiveHeader_t` *pfdir)

Encodes a CFDP file directive header block.
- void `CF_CFDP_EncodeLV` (`CF_EncoderState_t` *state, `CF_Logical_Lv_t` *pllv)

Encodes a single CFDP Length+Value (LV) pair.
- void `CF_CFDP_EncodeTLV` (`CF_EncoderState_t` *state, `CF_Logical_Tlv_t` *pltlv)

Encodes a single CFDP Type+Length+Value (TLV) tuple.
- void `CF_CFDP_EncodeSegmentRequest` (`CF_EncoderState_t` *state, `CF_Logical_SegmentRequest_t` *plseg)

Encodes a single CFDP Segment Request block.
- void `CF_CFDP_EncodeAllTlv` (`CF_EncoderState_t` *state, `CF_Logical_TlvList_t` *pltlv)

Encodes a list of CFDP Type+Length+Value tuples.
- void `CF_CFDP_EncodeAllSegments` (`CF_EncoderState_t` *state, `CF_Logical_SegmentList_t` *plseg)

Encodes a list of CFDP Segment Request blocks.
- void `CF_CFDP_EncodeMd` (`CF_EncoderState_t` *state, `CF_Logical_PduMd_t` *plmd)

Encodes a CFDP Metadata (MD) header block.
- void `CF_CFDP_EncodeFileDataHeader` (`CF_EncoderState_t` *state, bool with_meta, `CF_Logical_PduFileDataHeader_t` *plfd)

Encodes a CFDP File Data (FD) header block.
- void `CF_CFDP_EncodeEof` (`CF_EncoderState_t` *state, `CF_Logical_PduEof_t` *pleof)

Encodes a CFDP End-of-File (EOF) header block.
- void `CF_CFDP_EncodeFin` (`CF_EncoderState_t` *state, `CF_Logical_PduFin_t` *plfin)

Encodes a CFDP Final (FIN) header block.
- void `CF_CFDP_EncodeAck` (`CF_EncoderState_t` *state, `CF_Logical_PduAck_t` *plack)

Encodes a CFDP Acknowledge (ACK) header block.
- void `CF_CFDP_EncodeNak` (`CF_EncoderState_t` *state, `CF_Logical_PduNak_t` *plnak)

Encodes a CFDP Non-Acknowledge (NAK) header block.
- void `CF_CFDP_EncodeCrc` (`CF_EncoderState_t` *state, `uint32` *plcrc)

Encodes a CFDP CRC/Checksum.
- `CFE_Status_t CF_CFDP_DecodeHeader` (`CF_DecoderState_t` *state, `CF_Logical_PduHeader_t` *plh)

Decodes a CFDP base PDU header.
- void `CF_CFDP_DecodeFileDirectiveHeader` (`CF_DecoderState_t` *state, `CF_Logical_PduFileDirectiveHeader_t` *pfdir)

Decodes a CFDP file directive header block.
- void `CF_CFDP_DecodeLV` (`CF_DecoderState_t` *state, `CF_Logical_Lv_t` *pllv)

Decodes a single CFDP Length+Value (LV) pair.
- void `CF_CFDP_DecodeTLV` (`CF_DecoderState_t` *state, `CF_Logical_Tlv_t` *pltlv)

Decodes a single CFDP Type+Length+Value (TLV) tuple.
- void `CF_CFDP_DecodeSegmentRequest` (`CF_DecoderState_t` *state, `CF_Logical_SegmentRequest_t` *plseg)

Decodes a single CFDP Segment Request block.
- void `CF_CFDP_DecodeAllTlv` (`CF_DecoderState_t` *state, `CF_Logical_TlvList_t` *pltlv, `uint8` limit)

Decodes a list of CFDP Type+Length+Value tuples.
- void `CF_CFDP_DecodeAllSegments` (`CF_DecoderState_t` *state, `CF_Logical_SegmentList_t` *plseg, `uint8` limit)

Decodes a list of CFDP Segment Request blocks.
- void `CF_CFDP_DecodeMd` (`CF_DecoderState_t` *state, `CF_Logical_PduMd_t` *plmd)

Decodes a CFDP Metadata (MD) header block.
- void `CF_CFDP_DecodeFileDataHeader` (`CF_DecoderState_t` *state, bool with_meta, `CF_Logical_PduFileDataHeader_t` *plfd)

Decodes a CFDP File Data (FD) header block.

Decodes a CFDP File Data (FD) header block.

- void [CF_CFDP_DecodeEof](#) (CF_DecoderState_t *state, CF_Logical_PduEof_t *pleof)
Decodes a CFDP End-of-File (EOF) header block.
- void [CF_CFDP_DecodeFin](#) (CF_DecoderState_t *state, CF_Logical_PduFin_t *plfin)
Decodes a CFDP Final (FIN) header block.
- void [CF_CFDP_DecodeAck](#) (CF_DecoderState_t *state, CF_Logical_PduAck_t *plack)
Decodes a CFDP Acknowledge (ACK) header block.
- void [CF_CFDP_DecodeNak](#) (CF_DecoderState_t *state, CF_Logical_PduNak_t *plnak)
Decodes a CFDP Non-Acknowledge (NAK) header block.
- void [CF_CFDP_DecodeCrc](#) (CF_DecoderState_t *state, uint32 *plcrc)
Decodes a CFDP CRC/Checksum.

12.40.1 Detailed Description

CFDP protocol data structure encode/decode API declarations

12.40.2 Macro Definition Documentation

12.40.2.1 CF_CODEC_GET_POSITION #define CF_CODEC_GET_POSITION(
 s) ([CF_CFDP_CodecGetPosition](#)(&((s)->codec_state)))

Macro wrapper around [CF_CFDP_CodecGetPosition\(\)](#)

Checks the position of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF_EncoderState_t or CF_DecoderState_t object.

Parameters

s	Encoder or Decoder state
---	--------------------------

Definition at line 268 of file cf_codec.h.

12.40.2.2 CF_CODEC_GET_REMAIN #define CF_CODEC_GET_REMAIN(
 s) ([CF_CFDP_CodecGetRemain](#)(&((s)->codec_state)))

Macro wrapper around [CF_CFDP_CodecGetRemain\(\)](#)

Checks the remainder of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF_EncoderState_t or CF_DecoderState_t object.

Parameters

s	Encoder or Decoder state
---	--------------------------

Definition at line 280 of file cf_codec.h.

12.40.2.3 CF_CODEC_GET_SIZE #define CF_CODEC_GET_SIZE(
 s) ([CF_CFDP_CodecGetSize](#)(&((s)->codec_state)))

Macro wrapper around [CF_CFDP_CodecGetSize\(\)](#)

Checks the size of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF_EncoderState_t or CF_DecoderState_t object.

Parameters

<code>s</code>	Encoder or Decoder state
----------------	--------------------------

Definition at line 292 of file cf_codec.h.

12.40.2.4 CF_CODEC_IS_OK #define CF_CODEC_IS_OK(
`s`) ([CF_CFDP_CodecIsOK](#)(&((`s`)>codec_state)))

Macro wrapper around [CF_CFDP_CodecIsOK\(\)](#)

Checks the state of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF_EncoderState_t or CF_DecoderState_t object.

Parameters

<code>s</code>	Encoder or Decoder state
----------------	--------------------------

Definition at line 244 of file cf_codec.h.

12.40.2.5 CF_CODEC_SET_DONE #define CF_CODEC_SET_DONE(
`s`) ([CF_CFDP_CodecSetDone](#)(&((`s`)>codec_state)))

Macro wrapper around [CF_CFDP_CodecSetDone\(\)](#)

Sets the state of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF_EncoderState_t or CF_DecoderState_t object.

Parameters

<code>s</code>	Encoder or Decoder state
----------------	--------------------------

Definition at line 256 of file cf_codec.h.

12.40.2.6 CF_DECODE_FIXED_CHUNK #define CF_DECODE_FIXED_CHUNK(
`state`,
`type`) ((const type *)[CF_CFDP_DoDecodeChunk](#)(`state`, sizeof(type)))

Macro to decode a block of a given CFDP type into a PDU.

This is a wrapper around [CF_CFDP_DoDecodeChunk\(\)](#) to encode the given data type, rather than a generic size. The sizeof() the type should reflect the *encoded* size within the PDU. Specifically, this must only be used with the "CFDP" data types which are specifically designed to match the binary layout of the CFDP-defined header structures.

Parameters

<code>state</code>	Decoder state object
<code>type</code>	Data type to decode, from cf_cfdp_pdu.h

Returns

Pointer to block, if successful

Return values

<code>NULL</code>	if not successful (no space or other error).
-------------------	--

Definition at line 232 of file cf_codec.h.

12.40.2.7 CF_ENCODE_FIXED_CHUNK `#define CF_ENCODE_FIXED_CHUNK (`
`state,`
`type) ((type *)CF_CFDP_DoEncodeChunk(state, sizeof(type)))`

Macro to encode a block of a given CFDP type into a PDU.

This is a wrapper around [CF_CFDP_DoEncodeChunk\(\)](#) to encode the given data type, rather than a generic size. The `sizeof()` the type should reflect the *encoded* size within the PDU. Specifically, this must only be used with the "CFDP" data types which are specifically designed to match the binary layout of the CFDP-defined header structures.

Parameters

<code>state</code>	Encoder state object
<code>type</code>	Data type to encode, from cf_cfdp_pdu.h

Returns

Pointer to block, if successful

Return values

<code>NULL</code>	if not successful (no space or other error).
-------------------	--

Definition at line 215 of file cf_codec.h.

12.40.3 Typedef Documentation

12.40.3.1 CF_CodecState_t `typedef struct CF_CodecState CF_CodecState_t`

Tracks the current state of an encode or decode operation.

This encapsulates the common state between encode and decode

12.40.3.2 CF_DecoderState_t `typedef struct CF_DecoderState CF_DecoderState_t`

Current state of a decode operation.

State structure for decodes

12.40.3.3 CF_EncoderState_t `typedef struct CF_EncoderState CF_EncoderState_t`

Current state of an encode operation.

State structure for encodes

12.40.4 Function Documentation

```
12.40.4.1 CF_CFDP_CodecCheckSize() bool CF_CFDP_CodecCheckSize (
    CF_CodecState_t * state,
    size_t chunksize )
```

Advances the position by the indicated size, confirming the block will fit into the PDU.

On encode, this confirms there is enough available space to hold a block of the indicated size. On decode, this confirms that decoding the indicated number of bytes will not read beyond the end of data.

If true, then the current position/offset is advanced by the indicated number of bytes. If false, then the error flag is set, so that future calls to CF_CFDP_CodecIsOK will also return false.

Note

The error flag is sticky, meaning that if any encode/decode operation fails, all future encode/decode requests on the same state will also fail. Each encode/decode step must check the flag, and skip the operation if it is false. Reporting the error can be deferred to the final stage, and only done once.

Parameters

<i>state</i>	Encoder/Decoder common state
<i>chunksize</i>	Size of next block to encode/decode

Return values

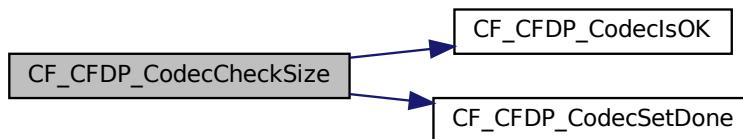
<i>true</i>	If encode/decode is possible, enough space exists
<i>false</i>	If encode/decode is not possible, not enough space or prior error occurred

Definition at line 269 of file cf_codec.c.

References CF_CFDP_CodecIsOK(), CF_CFDP_CodecSetDone(), CF_CodecState::max_size, and CF_CodecState::next_offset.

Referenced by CF_CFDP_DoDecodeChunk(), and CF_CFDP_DoEncodeChunk().

Here is the call graph for this function:



```
12.40.4.2 CF_CFDP_CodecGetPosition() static size_t CF_CFDP_CodecGetPosition (
    const CF_CodecState_t * state ) [inline], [static]
```

Obtains the current position/offset within the PDU.

Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Returns

Current offset in PDU

Definition at line 107 of file cf_codec.h.

References CF_CodecState::next_offset.

Referenced by CF_CFDP_DoDecodeChunk(), and CF_CFDP_DoEncodeChunk().

12.40.4.3 CF_CFDP_CodecGetRemain() static size_t CF_CFDP_CodecGetRemain (

```
const CF_CodecState_t * state ) [inline], [static]
```

Obtains the remaining size of the PDU being encoded/decoded.

Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Returns

Remaining size of PDU

Definition at line 131 of file cf_codec.h.

References CF_CodecState::max_size, and CF_CodecState::next_offset.

12.40.4.4 CF_CFDP_CodecGetSize() static size_t CF_CFDP_CodecGetSize (

```
const CF_CodecState_t * state ) [inline], [static]
```

Obtains the maximum size of the PDU being encoded/decoded.

Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Returns

Maximum size of PDU

Definition at line 119 of file cf_codec.h.

References CF_CodecState::max_size.

12.40.4.5 CF_CFDP_CodecIsOK() static bool CF_CFDP_CodecIsOK (

```
const CF_CodecState_t * state ) [inline], [static]
```

Checks if the codec is currently valid or not.

Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Return values

<i>true</i>	If encoder/decoder is still valid, has not reached end of PDU
<i>false</i>	If encoder/decoder is not valid, has reached end of PDU or an error occurred

Definition at line 82 of file cf_codec.h.

References CF_CodecState::is_valid.

Referenced by CF_CFDP_CodecCheckSize().

12.40.4.6 CF_CFDP_CodecReset() static void CF_CFDP_CodecReset (
`CF_CodecState_t * state,`
`size_t max_size) [inline], [static]`

Resets a codec state.

Parameters

<i>state</i>	Encoder/Decoder common state
<i>max_size</i>	Maximum size of PDU

Definition at line 143 of file cf_codec.h.

References CF_CodecState::is_valid, CF_CodecState::max_size, and CF_CodecState::next_offset.

Referenced by CF_CFDP_DecodeStart(), and CF_CFDP_EncodeStart().

12.40.4.7 CF_CFDP_CodecSetDone() static void CF_CFDP_CodecSetDone (
`CF_CodecState_t * state) [inline], [static]`

Sets a codec to the "done" state.

This may mean end of PDU data is reached, or that an error occurred

Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Definition at line 95 of file cf_codec.h.

References CF_CodecState::is_valid.

Referenced by CF_CFDP_CodecCheckSize(), CF_CFDP_DecodeStart(), and CF_CFDP_EncodeStart().

12.40.4.8 CF_CFDP_DecodeAck() void CF_CFDP_DecodeAck (
`CF_DecoderState_t * state,`
`CF_Logical_PduAck_t * plack)`

Decodes a CFDP Acknowledge (ACK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 1043 of file cf_codec.c.

References CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::cc, CF_CFDP_PduAck::cc_and_transaction_status, CF_CFDP_PduAck_CC, CF_CFDP_PduAck_DIR_CODE, CF_CFDP_PduAck_DIR_SUBTYPE_CODE, CF_CFDP_PduAck_TRANSACTION_STATUS, CF_DECODE_FIXED_CHUNK, CF_CFDP_PduAck::directive_and_subtype_code, FGV, and CF_Logical_PduAck::txn_status.

Referenced by CF_CFDP_RecvAck().

```
12.40.4.9 CF_CFDP_DecodeAllSegments() void CF_CFDP_DecodeAllSegments (
    CF_DecoderState_t * state,
    CF_Logical_SegmentList_t * plseg,
    uint8 limit )
```

Decodes a list of CFDP Segment Request blocks.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

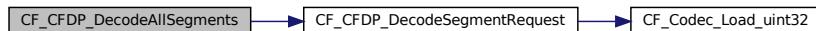
<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data
<i>limit</i>	Maximum number of Segment Request objects to decode

Definition at line 1119 of file cf_codec.c.

References CF_CFDP_DecodeSegmentRequest(), CF_CODEC_GET_REMAIN, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_PDU_MAX_SEGMENTS, CF_Logical_SegmentList::num_segments, and CF_Logical_SegmentList::segments.

Referenced by CF_CFDP_DecodeNak().

Here is the call graph for this function:



```
12.40.4.10 CF_CFDP_DecodeAllTlv() void CF_CFDP_DecodeAllTlv (
    CF_DecoderState_t * state,
    CF_Logical_TlvList_t * pltlv,
    uint8 limit )
```

Decodes a list of CFDP Type+Length+Value tuples.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

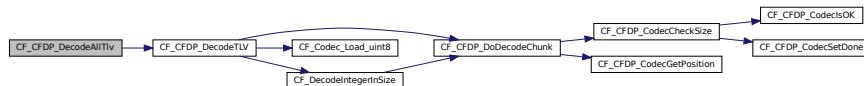
<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data
<i>limit</i>	Maximum number of TLV objects to decode

Definition at line 1084 of file cf_codec.c.

References CF_CFDP_DecodeTLV(), CF_CODEC_GET_REMAIN, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_PDU_MAX_TLV, CF_Logical_TlvList::num_tlv, and CF_Logical_TlvList::tlv.

Referenced by CF_CFDP_DecodeEof(), and CF_CFDP_DecodeFin().

Here is the call graph for this function:



12.40.4.11 CF_CFDP_DecodeCrc() `void CF_CFDP_DecodeCrc (`

```
CF_DecoderState_t * state,
uint32 * plcrc )
```

Decodes a CFDP CRC/Checksum.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

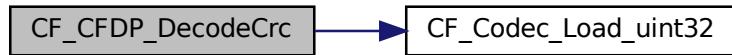
Parameters

<code>state</code>	Decoder state object
<code>plcrc</code>	Pointer to logical CRC value

Definition at line 984 of file cf_codec.c.

References CF_Codec_Load_uint32(), and CF_DECODE_FIXED_CHUNK.

Here is the call graph for this function:



12.40.4.12 CF_CFDP_DecodeEof() `void CF_CFDP_DecodeEof (`

```
CF_DecoderState_t * state,
CF_Logical_PduEof_t * pleof )
```

Decodes a CFDP End-of-File (EOF) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

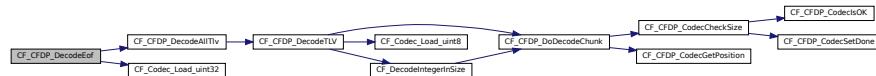
<code>state</code>	Decoder state object
<code>pleof</code>	Pointer to logical PDU EOF header data

Definition at line 1001 of file cf_codec.c.

References CF_Logical_PduEof::cc, CF_CFDP_PduEof::cc, CF_CFDP_DecodeAllTlv(), CF_CFDP_PduEof_FLAGS←_CC, CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_TLV, CF_Logical_PduEof::crc, CF_CFDP_PduEof::crc, FGV, CF_Logical_PduEof::size, CF_CFDP_PduEof::size, and CF_Logical_PduEof::tlv_list.

Referenced by CF_CFDP_RecvEof().

Here is the call graph for this function:



12.40.4.13 CF_CFDP_DecodeFileDataHeader()

```
void CF_CFDP_DecodeFileDataHeader (
    CF_DecoderState_t * state,
    bool with_meta,
    CF_Logical_PduFileDataHeader_t * plfd )
```

Decodes a CFDP File Data (FD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

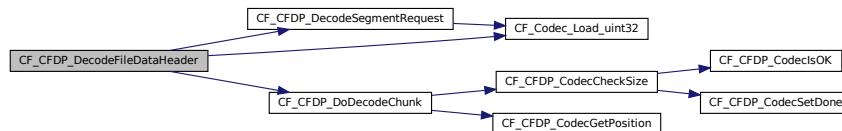
<i>state</i>	Decoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>plfd</i>	Pointer to logical PDU file header data

Definition at line 922 of file cf_codec.c.

References CF_CFDP_DecodeSegmentRequest(), CF_CFDP_DoDecodeChunk(), CF_CFDP_PduFileData_RECO←RD_CONTINUATION_STATE, CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH, CF_CODEC_GET_R←EMAIN, CF_CODEC_IS_OK, CF_Codec_Load_uint32(), CF_CODEC_SET_DONE, CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_SEGMENTS, CF_Logical_PduFileDataHeader::continuation_state, CF_Logical_PduFileDataHeader::data_len, CF_Logical_PduFileDataHeader::data_ptr, FGV, CF_Logical_SegmentList::num_segments, CF_Logical_PduFileDataHeader::offset, CF_CFDP_PduFileDataHeader::offset, CF_Logical_PduFileDataHeader::segment_list, and CF_Logical_SegmentList::segments.

Referenced by CF_CFDP_RecvFd().

Here is the call graph for this function:



12.40.4.14 CF_CFDP_DecodeFileDirectiveHeader()

```
CF_DecoderState_t * state,
CF_Logical_PduFileDirectiveHeader_t * pfdir )
```

Decodes a CFDP file directive header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 812 of file cf_codec.c.

References CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_Logical_PduFileDirectiveHeader::directive_code, and CF_CFDP_PduFileDirectiveHeader::directive_code.

Referenced by CF_CFDP_RecvPh().

Here is the call graph for this function:



12.40.4.15 CF_CFDP_DecodeFin() void CF_CFDP_DecodeFin (

```
CF_DecoderState_t * state,
CF_Logical_PduFin_t * plfin )
```

Decodes a CFDP Final (FIN) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>plfin</i>	Pointer to logical PDU FIN header data

Definition at line 1022 of file cf_codec.c.

References CF_Logical_PduFin::cc, CF_CFDP_DecodeAllTlv(), CF_CFDP_PduFin_FLAGS_CC, CF_CFDP_PduFin_FLAGS_DELIVERY_CODE, CF_CFDP_PduFin_FLAGS_FILE_STATUS, CF_DECODE_FIXED_CHUNK, CF_PDU_MAX_TLV, CF_Logical_PduFin::delivery_code, FGV, CF_Logical_PduFin::file_status, CF_CFDP_PduFin::flags, and CF_Logical_PduFin::tlv_list.

Referenced by CF_CFDP_RecvFin().

Here is the call graph for this function:



12.40.4.16 CF_CFDP_DecodeHeader() `CFE_Status_t CF_CFDP_DecodeHeader (`
`CF_DecoderState_t * state,`
`CF_Logical_PduHeader_t * plh)`

Decodes a CFDP base PDU header.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Note

On decode the entire base header is decoded in a single call, the size will be decoded like any other field.

Parameters

<code>state</code>	Decoder state object
<code>plh</code>	Pointer to logical PDU base header data

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CF_ERROR</code>	on error.

Definition at line 766 of file cf_codec.c.

References CF_CFDP_PduHeader_FLAGS_CRC, CF_CFDP_PduHeader_FLAGS_DIR, CF_CFDP_PduHeader_FLAGS_LARGEFILE, CF_CFDP_PduHeader_FLAGS_MODE, CF_CFDP_PduHeader_FLAGS_TYPE, CF_CFDP_PduHeader_FLAGS_VERSION, CF_CFDP_PduHeader_LENGTHS_ENTITY, CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE, CF_CODEC_GET_POSITION, CF_Codec_Load_uint16(), CF_DECODE_FIXED_CHUNK, CF_DecodeIntegerInSize(), CF_ERROR, CFE_SUCCESS, CF_Logical_PduHeader::crc_flag, CF_Logical_PduHeader::data_encoded_length, CF_Logical_PduHeader::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduHeader::eid_length, CF_CFDP_PduHeader::eid_tsn_lengths, FGV, CF_CFDP_PduHeader::flags, CF_Logical_PduHeader::header_encoded_length, CF_Logical_PduHeader::large_flag, CF_CFDP_PduHeader::length, CF_Logical_PduHeader::pdu_type, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_RecvPh().

Here is the call graph for this function:



```
12.40.4.17 CF_CFDP_DecodeLV() void CF_CFDP_DecodeLV (
    CF_DecoderState_t * state,
    CF_Logical_Lv_t * pllv )
```

Decodes a single CFDP Length+Value (LV) pair.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

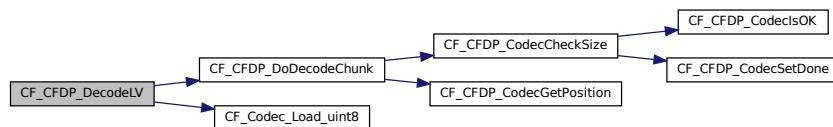
<i>state</i>	Decoder state object
<i>pllv</i>	Pointer to single logical PDU LV data

Definition at line 832 of file cf_codec.c.

References CF_CFDP_DoDecodeChunk(), CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_Logical_Lv::data_ptr, CF_Logical_Lv::length, and CF_CFDP_Lv::length.

Referenced by CF_CFDP_DecodeMd().

Here is the call graph for this function:



```
12.40.4.18 CF_CFDP_DecodeMd() void CF_CFDP_DecodeMd (
    CF_DecoderState_t * state,
    CF_Logical_PduMd_t * plmd )
```

Decodes a CFDP Metadata (MD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

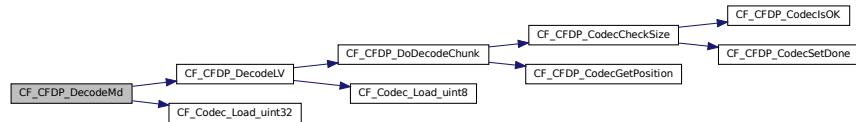
<i>state</i>	Decoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 899 of file cf_codec.c.

References CF_CFDP_DecodeLV(), CF_CFDP_PduMd_CHECKSUM_TYPE, CF_CFDP_PduMd_CLOSURE_REQUESTED, CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_Logical_PduMd::checksum_type, CF_Logical_PduMd::close_req, CF_Logical_PduMd::dest_filename, FGV, CF_CFDP_PduMd::segmentation_control, CF_Logical_PduMd::size, CF_CFDP_PduMd::size, and CF_Logical_PduMd::source_filename.

Referenced by CF_CFDP_RecvMd().

Here is the call graph for this function:



12.40.4.19 CF_CFDP_DecodeNak() `void CF_CFDP_DecodeNak (`
`CF_DecoderState_t * state,`
`CF_Logical_PduNak_t * plnak)`

Decodes a CFDP Non-Acknowledge (NAK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

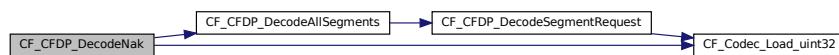
<code>state</code>	Decoder state object
<code>plnak</code>	Pointer to logical PDU NAK header data

Definition at line 1064 of file cf_codec.c.

References `CF_CFDP_DecodeAllSegments()`, `CF_Codec_Load_uint32()`, `CF_DECODE_FIXED_CHUNK`, `CF_PDU_MAX_SEGMENTS`, `CF_Logical_PduNak::scope_end`, `CF_CFDP_PduNak::scope_end`, `CF_Logical_PduNak::scope_start`, `CF_CFDP_PduNak::scope_start`, and `CF_Logical_PduNak::segment_list`.

Referenced by `CF_CFDP_RecvNak()`.

Here is the call graph for this function:



12.40.4.20 CF_CFDP_DecodeSegmentRequest() `void CF_CFDP_DecodeSegmentRequest (`
`CF_DecoderState_t * state,`
`CF_Logical_SegmentRequest_t * plseg)`

Decodes a single CFDP Segment Request block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<code>state</code>	Decoder state object
<code>plseg</code>	Pointer to single logical PDU segment request header data

Definition at line 881 of file cf_codec.c.

References CF_Codec_Load_uint32(), CF_DECODE_FIXED_CHUNK, CF_Logical_SegmentRequest::offset_end, CF_CFDP_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, and CF_CFDP_SegmentRequest::offset_start.

Referenced by CF_CFDP_DecodeAllSegments(), and CF_CFDP_DecodeFileDataHeader().

Here is the call graph for this function:



12.40.4.21 CF_CFDP_DecodeTLV()

```
void CF_CFDP_DecodeTLV (
    CF_DecoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Decodes a single CFDP Type+Length+Value (TLV) tuple.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

Parameters

<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV data

Definition at line 850 of file cf_codec.c.

References CF_CFDP_DoDecodeChunk(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_Codec_Load_uint8(), CF_DECODE_FIXED_CHUNK, CF_DecodeIntegerInSize(), CF_Logical_Tlv::data, CF_Logical_TlvData::data_ptr, CF_Logical_TlvData::eid, CF_CFDP_tlv::length, CF_Logical_Tlv::length, CF_CFDP_tlv::type, and CF_Logical_Tlv::type.

Referenced by CF_CFDP_DecodeAllTLV().

Here is the call graph for this function:



12.40.4.22 CF_CFDP_DoDecodeChunk()

```
const void* CF_CFDP_DoDecodeChunk (
    CF_DecoderState_t * state,
    size_t chunksize )
```

Decode a block of data from the PDU.

Deducts space for a block of the given size from the current PDU

Parameters

<i>state</i>	Decoder state object
<i>chunksize</i>	Size of block to decode

Returns

Pointer to block, if successful

Return values

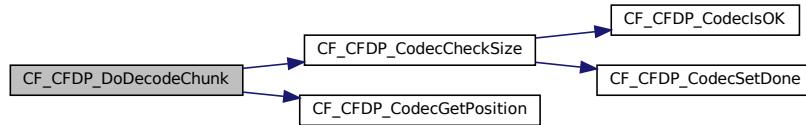
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 309 of file cf_codec.c.

References CF_DecoderState::base, CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetPosition(), and CF_DecoderState::codec_state.

Referenced by CF_CFDP_DecodeFileDataHeader(), CF_CFDP_DecodeLV(), CF_CFDP_DecodeTLV(), and CF_DecodeIntegerInSize().

Here is the call graph for this function:

**12.40.4.23 CF_CFDP_DoEncodeChunk()** `void* CF_CFDP_DoEncodeChunk (`

```
    CF_EncoderState_t * state,
    size_t chunksize )
```

Encode a block of data into the PDU.

Adds/Reserves space for a block of the given size in the current PDU

Parameters

<i>state</i>	Encoder state object
<i>chunksize</i>	Size of block to encode

Returns

Pointer to block, if successful

Return values

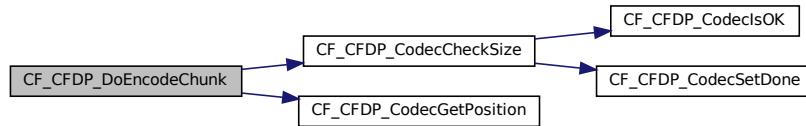
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 291 of file cf_codec.c.

References CF_EncoderState::base, CF_CFDP_CodecCheckSize(), CF_CFDP_CodecGetPosition(), and CF_EncoderState::codec_state.

Referenced by CF_CFDP_EncodeLV(), CF_CFDP_EncodeTLV(), CF_CFDP_S_SendFileData(), and CF_EncodeIntegerInSize().

Here is the call graph for this function:



12.40.4.24 CF_CFDP_EncodeAck() `void CF_CFDP_EncodeAck (`
 `CF_EncoderState_t * state,`
 `CF_Logical_PduAck_t * plack)`

Encodes a CFDP Acknowledge (ACK) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<code>state</code>	Encoder state object
<code>plack</code>	Pointer to logical PDU ACK header data

Definition at line 678 of file cf_codec.c.

References CF_Logical_PduAck::ack_directive_code, CF_Logical_PduAck::ack_subtype_code, CF_Logical_PduAck::cc, CF_CFDP_PduAck::cc_and_transaction_status, CF_CFDP_PduAck_CC, CF_CFDP_PduAck_DIR_CODE, CF_CFDP_PduAck_DIR_SUBTYPE_CODE, CF_CFDP_PduAck_TRANSACTION_STATUS, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_CFDP_PduAck::directive_and_subtype_code, FSV, and CF_Logical_PduAck::tx_status.

Referenced by CF_CFDP_SendAck().

Here is the call graph for this function:



12.40.4.25 CF_CFDP_EncodeAllSegments() `void CF_CFDP_EncodeAllSegments (`

```
CF_EncoderState_t * state,
CF_Logical_SegmentList_t * plseg )
```

Encodes a list of CFDP Segment Request blocks.

This invokes [CF_CFDP_EncodeSegmentRequest\(\)](#) for all segments in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

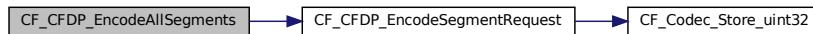
<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data

Definition at line 557 of file cf_codec.c.

References [CF_CFDP_EncodeSegmentRequest\(\)](#), [CF_CODEC_IS_OK](#), [CF_Logical_SegmentList::num_segments](#), and [CF_Logical_SegmentList::segments](#).

Referenced by [CF_CFDP_EncodeFileDataHeader\(\)](#), and [CF_CFDP_EncodeNak\(\)](#).

Here is the call graph for this function:



12.40.4.26 CF_CFDP_EncodeAllTlv() void CF_CFDP_EncodeAllTlv (

```
CF_EncoderState_t * state,
CF_Logical_TlvList_t * pltlv )
```

Encodes a list of CFDP Type+Length+Value tuples.

This invokes [CF_CFDP_EncodeTLV\(\)](#) for all TLV values in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

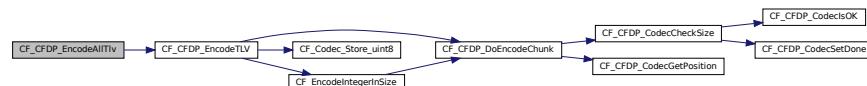
Parameters

<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data

Definition at line 541 of file cf_codec.c.

References [CF_CFDP_EncodeTLV\(\)](#), [CF_CODEC_IS_OK](#), [CF_Logical_TlvList::num_tlv](#), and [CF_Logical_TlvList::tlv](#). Referenced by [CF_CFDP_EncodeEof\(\)](#), and [CF_CFDP_EncodeFin\(\)](#).

Here is the call graph for this function:



```
12.40.4.27 CF_CFDP_EncodeCrc() void CF_CFDP_EncodeCrc (
    CF_EncoderState_t * state,
    uint32 * plcrc )
```

Encodes a CFDP CRC/Checksum.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<i>state</i>	Encoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 721 of file cf_codec.c.

References CF_Codec_Store_uint32(), and CF_ENCODE_FIXED_CHUNK.

Here is the call graph for this function:



```
12.40.4.28 CF_CFDP_EncodeEof() void CF_CFDP_EncodeEof (
    CF_EncoderState_t * state,
    CF_Logical_PduEof_t * pleof )
```

Encodes a CFDP End-of-File (EOF) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any TLV values which are indicated in the logical data structure

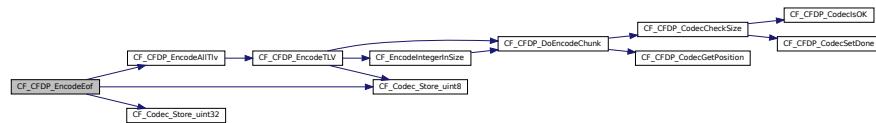
Parameters

<i>state</i>	Encoder state object
<i>pleof</i>	Pointer to logical PDU EOF header data

Definition at line 634 of file cf_codec.c.

References CF_Logical_PduEof::cc, CF_CFDP_PduEof::cc, CF_CFDP_EncodeAllTlv(), CF_CFDP_PduEof_FLAGS←_CC, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduEof::crc, CF_CFDP_PduEof::crc, FSV, CF_Logical_PduEof::size, CF_CFDP_PduEof::size, and CF_Logical_PduEof::tlv_list.
Referenced by CF_CFDP_SendEof().

Here is the call graph for this function:



12.40.4.29 CF_CFDP_EncodeFileDataHeader() void CF_CFDP_EncodeFileDataHeader (
`CF_EncoderState_t * state,`
`bool with_meta,`
`CF_Logical_PduFileDataHeader_t * plfd)`

Encodes a CFDP File Data (FD) header block.

This only encodes the FD header fields, specifically the data offset (required) and any metadata fields, if indicated. This does *not* encode any actual file data.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

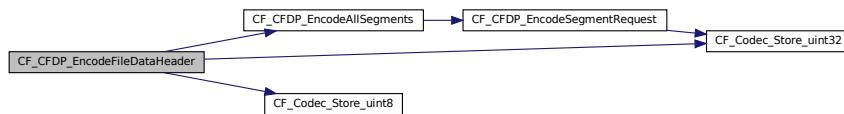
<code>state</code>	Encoder state object
<code>with_meta</code>	Whether to include optional continuation and segment request fields (always false currently)
<code>plfd</code>	Pointer to logical PDU file header data

Definition at line 597 of file cf_codec.c.

References CF_CFDP_EncodeAllSegments(), CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE, CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_E_NCODE_FIXED_CHUNK, CF_Logical_PduFileDataHeader::continuation_state, FSV, CF_Logical_SegmentList::num_segments, CF_Logical_PduFileDataHeader::offset, CF_CFDP_PduFileDataHeader::offset, and CF_Logical_PduFileDataHeader::segment_list.

Referenced by CF_CFDP_S_SendFileData().

Here is the call graph for this function:



12.40.4.30 CF_CFDP_EncodeFileDirectiveHeader() void CF_CFDP_EncodeFileDirectiveHeader (
`CF_EncoderState_t * state,`
`CF_Logical_PduFileDirectiveHeader_t * pfdir)`

Encodes a CFDP file directive header block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

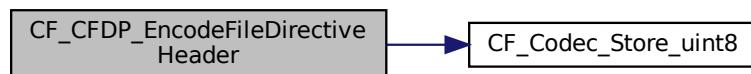
<i>state</i>	Encoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 437 of file cf_codec.c.

References CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduFileDirectiveHeader::directive_code, and CF_CFDP_PduFileDirectiveHeader::directive_code.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.40.4.31 CF_CFDP_EncodeFin() void CF_CFDP_EncodeFin (

```

    CF_EncoderState_t * state,
    CF_Logical_PduFin_t * plfin )

```

Encodes a CFDP Final (FIN) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any TLV values which are indicated in the logical data structure

Parameters

<i>state</i>	Encoder state object
<i>plfin</i>	Pointer to logical PDU FIN header data

Definition at line 656 of file cf_codec.c.

References CF_Logical_PduFin::cc, CF_CFDP_EncodeAllTlv(), CF_CFDP_PduFin_FLAGS_CC, CF_CFDP_PduFin::FLAGS_DELIVERY_CODE, CF_CFDP_PduFin_FLAGS_FILE_STATUS, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduFin::delivery_code, CF_Logical_PduFin::file_status, CF_CFDP_PduFin::flags, FSV, and CF_Logical_PduFin::tlv_list.

Referenced by CF_CFDP_SendFin().

Here is the call graph for this function:



```
12.40.4.32 CF_CFDP_EncodeHeaderFinalSize() void CF_CFDP_EncodeHeaderFinalSize (
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Updates an already-encoded PDU base header block with the final PDU size.

This function encodes the "data_encoded_length" field from the logical PDU structure into the encoded header block. The PDU will also be closed (set done) to indicate that no more data should be added.

Note

Unlike other encode operations, this function does not add any new blocks to the PDU. It only updates the already-encoded block at the beginning of the PDU, which must have been done by a prior call to [CF_CFDP_EncodeHeaderWithoutSize\(\)](#).

See also

[CF_CFDP_EncodeHeaderWithoutSize\(\)](#) for initially encoding the PDU header block

Parameters

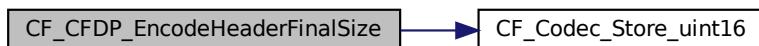
<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 407 of file cf_codec.c.

References CF_EncoderState::base, CF_CODEC_GET_POSITION, CF_CODEC_IS_OK, CF_CODEC_SET_DONE, CF_Codec_Store_uint16(), CF_Logical_PduHeader::data_encoded_length, and CF_CFDP_PduHeader::length.

Referenced by CF_CFDP_SetPduLength().

Here is the call graph for this function:



```
12.40.4.33 CF_CFDP_EncodeHeaderWithoutSize() void CF_CFDP_EncodeHeaderWithoutSize (
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Encodes a CFDP PDU base header block, bypassing the size field.

On transmit side, the common/base header must be encoded in two parts, to deal with the "total_size" field. The initial encoding of the basic fields is done as soon as it is known that a PDU of this type needs to be sent, but the total size may not be yet known, as it depends on the remainder of encoding and any additional data that might get added to the variable length sections.

This function encodes all base header fields *except* total length. There is a separate function later to update the total_length to the correct value once the remainder of encoding is done. Luckily, the total_length is in the first fixed position binary blob so it is easy to update later.

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

See also

[CF_CFDP_EncodeHeaderFinalSize\(\)](#) for updating the length field once it is known

Parameters

<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 371 of file cf_codec.c.

References CF_CFDP_PduHeader_FLAGS_DIR, CF_CFDP_PduHeader_FLAGS_MODE, CF_CFDP_PduHeader_FLAGS_TYPE, CF_CFDP_PduHeader_FLAGS_VERSION, CF_CFDP_PduHeader_LENGTHS_ENTITY, CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE, CF_CODEC_GET_POSITION, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_EncodeIntegerInSize(), CF_Logical_PduHeader::destination_eid, CF_Logical_PduHeader::direction, CF_Logical_PduHeader::eid_length, CF_CFDP_PduHeader::eid_tsn_lengths, CF_CFDP_PduHeader::flags, FSV, CF_Logical_PduHeader::header_encoded_length, CF_Logical_PduHeader::pdu_type, CF_Logical_PduHeader::sequence_num, CF_Logical_PduHeader::source_eid, CF_Logical_PduHeader::txm_mode, CF_Logical_PduHeader::txn_seq_length, and CF_Logical_PduHeader::version.

Referenced by CF_CFDP_ConstructPduHeader().

Here is the call graph for this function:



12.40.4.34 CF_CFDP_EncodeLV() void CF_CFDP_EncodeLV (
CF_EncoderState_t * *state*,
CF_Logical_Lv_t * *pllv*)

Encodes a single CFDP Length+Value (LV) pair.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

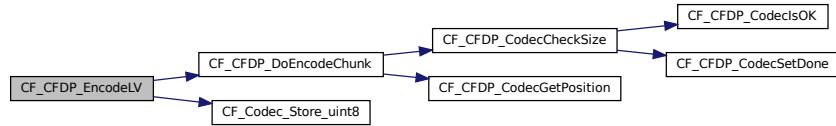
<i>state</i>	Encoder state object
<i>pllv</i>	Pointer to logical PDU LV header data

Definition at line 455 of file cf_codec.c.

References CF_CFDP_DoEncodeChunk(), CF_CODEC_SET_DONE, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_Lv::data_ptr, CF_Logical_Lv::length, and CF_CFDP_Lv::length.

Referenced by CF_CFDP_EncodeMd().

Here is the call graph for this function:



12.40.4.35 CF_CFDP_EncodeMd() void CF_CFDP_EncodeMd (

- CF_EncoderState_t * state,
- CF_Logical_PduMd_t * plmd)

Encodes a CFDP Metadata (MD) header block.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes the LV pairs for source and destination file names, which are logically part of the overall MD block.

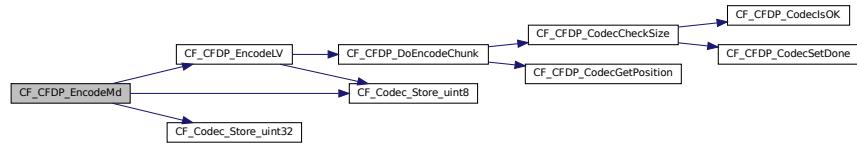
Parameters

<i>state</i>	Encoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 573 of file cf_codec.c.

References CF_CFDP_EncodeLV(), CF_CFDP_PduMd_CHECKSUM_TYPE, CF_CFDP_PduMd_CLOSURE_REQ, UESTED, CF_Codec_Store_uint32(), CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduMd::checksum_type, CF_Logical_PduMd::close_req, CF_Logical_PduMd::dest_filename, FSV, CF_CFDP_PduMd::segmentation_control, CF_Logical_PduMd::size, CF_CFDP_PduMd::size, and CF_Logical_PduMd::source_filename.
Referenced by CF_CFDP_SendMd().

Here is the call graph for this function:



12.40.4.36 CF_CFDP_EncodeNak() void CF_CFDP_EncodeNak (

- CF_EncoderState_t * state,
- CF_Logical_PduNak_t * plnak)

Encodes a CFDP Non-Acknowledge (NAK) header block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

this encode includes any Segment Request values which are indicated in the logical data structure

Parameters

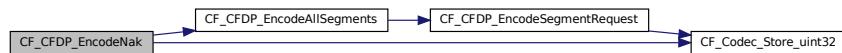
<i>state</i>	Encoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 701 of file cf_codec.c.

References CF_CFDP_EncodeAllSegments(), CF_Codec_Store_uint32(), CF_ENCODE_FIXED_CHUNK, CF_Logical_PduNak::scope_end, CF_CFDP_PduNak::scope_end, CF_Logical_PduNak::scope_start, CF_CFDP_PduNak::scope_start, and CF_Logical_PduNak::segment_list.

Referenced by CF_CFDP_SendNak().

Here is the call graph for this function:



12.40.4.37 CF_CFDP_EncodeSegmentRequest()

```
void CF_CFDP_EncodeSegmentRequest (
    CF_EncoderState_t * state,
    CF_Logical_SegmentRequest_t * plseg )
```

Encodes a single CFDP Segment Request block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Parameters

<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to single logical PDU segment request header data

Definition at line 523 of file cf_codec.c.

References CF_Codec_Store_uint32(), CF_ENCODE_FIXED_CHUNK, CF_Logical_SegmentRequest::offset_end, CF_CFDP_SegmentRequest::offset_end, CF_Logical_SegmentRequest::offset_start, and CF_CFDP_SegmentRequest::offset_start.

Referenced by CF_CFDP_EncodeAllSegments().

Here is the call graph for this function:



```
12.40.4.38 CF_CFDP_EncodeTLV() void CF_CFDP_EncodeTLV (
    CF_EncoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Encodes a single CFDP Type+Length+Value (TLV) tuple.

The data in the logical header will be appended to the encoded PDU at the current position
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

Note

Only the CF_CFDP_TLV_TYPE_ENTITY_ID TLV type is currently supported by this function, but other TLV types may be added in future versions as needed.

Parameters

<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV header data

Definition at line 485 of file cf_codec.c.

References CF_CFDP_DoEncodeChunk(), CF_CFDP_TLV_TYPE_ENTITY_ID, CF_CODEC_SET_DONE, CF_Codec_Store_uint8(), CF_ENCODE_FIXED_CHUNK, CF_EncodeIntegerInSize(), CF_Logical_Tlv::data, CF_Logical_TlvData::data_ptr, CF_Logical_TlvData::eid, CF_Logical_Tlv::length, CF_CFDP_tlv::length, CF_CFDP_tlv::type, and CF_Logical_Tlv::type.

Referenced by CF_CFDP_EncodeAllTlv().

Here is the call graph for this function:



```
12.40.4.39 CF_CFDP_GetValueEncodedSize() uint8 CF_CFDP_GetValueEncodedSize (
    uint64 Value )
```

Gets the minimum number of octets that the given integer may be encoded in.

Based on the integer value, this computes the minimum number of bytes that must be allocated to that integer within a CFDP PDU. This is typically used for entity IDs and sequence numbers, where CFDP does not specify a specific size for these items. They may be encoded between 1 and 8 bytes, depending on the actual value is.

Parameters

<i>Value</i>	Integer value that needs to be encoded
--------------	--

Returns

Minimum number of bytes that the value requires (between 1 and 8, inclusive)

Definition at line 327 of file cf_codec.c.

Referenced by CF_CFDP_AppendTlv(), and CF_CFDP_ConstructPduHeader().

```
12.40.4.40 CF_DecodeIntegerInSize() uint64 CF_DecodeIntegerInSize (
    CF_DecoderState_t * state,
    uint8 decode_size )
```

Decodes an integer value from the specified number of octets.

This decodes an integer value in the specified number of octets. The actual number of octets must be determined using another field in the PDU before calling this function.

Warning

This function will decode exactly the given number of octets. If this does not match actual encoded size, the return value will be wrong, and it will likely also throw off the decoding of any fields that follow this one.

See also

[CF_EncodeIntegerInSize\(\)](#) for the inverse operation

Parameters

<i>state</i>	Encoder state object
<i>decode_size</i>	Number of octets that the value is encoded in (between 1 and 8, inclusive)

Returns

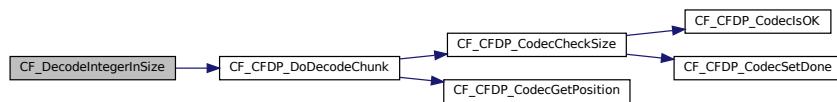
Decoded value

Definition at line 738 of file cf_codec.c.

References [CF_CFDP_DoDecodeChunk\(\)](#).

Referenced by [CF_CFDP_DecodeHeader\(\)](#), and [CF_CFDP_DecodeTLV\(\)](#).

Here is the call graph for this function:



```
12.40.4.41 CF_EncodeIntegerInSize() void CF_EncodeIntegerInSize (
    CF_EncoderState_t * state,
    uint64 value,
    uint8 encode_size )
```

Encodes the given integer value in the given number of octets.

This encodes an integer value in the specified number of octets. Use [CF_CFDP_GetValueEncodedSize\(\)](#) to determine the minimum number of octets required for a given value. Using more than the minimum is OK, but will consume extra bytes.

Warning

This function does not error check the encode_size parameter, and will encode the size given, even if it results in an invalid value. Using fewer octets than the minimum reported by [CF_CFDP_GetValueEncodedSize\(\)](#) will likely result in incorrect decoding at the receiver.

See also

[CF_DecodeIntegerInSize\(\)](#) for the inverse operation

Parameters

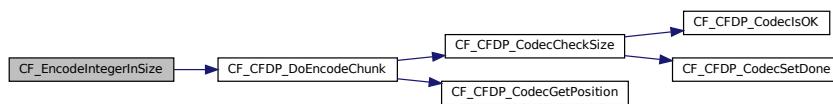
<i>state</i>	Encoder state object
<i>value</i>	Integer value that needs to be encoded
<i>encode_size</i>	Number of octets to encode the value in (between 1 and 8, inclusive)

Definition at line 346 of file cf_codec.c.

References [CF_CFDP_DoEncodeChunk\(\)](#).

Referenced by [CF_CFDP_EncodeHeaderWithoutSize\(\)](#), and [CF_CFDP_EncodeTLV\(\)](#).

Here is the call graph for this function:



12.41 apps/cf/fsw/src(cf_crc.c) File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_crc.h"
#include <string.h>
```

Functions

- void [CF_CRC_Start \(CF_Crc_t *crc\)](#)
Start a CRC streamable digest.
- void [CF_CRC_Digest \(CF_Crc_t *crc, const uint8 *data, size_t len\)](#)
Digest a chunk for CRC calculation.
- void [CF_CRC_Finalize \(CF_Crc_t *crc\)](#)
Finalize a CRC calculation.

12.41.1 Detailed Description

The CF Application CRC calculation source file

This is a streaming CRC calculator. Data can all be given at once for a result or it can trickle in.

This file is intended to be generic and usable by other apps.

12.41.2 Function Documentation

12.41.2.1 CF_CRC_Digest() `void CF_CRC_Digest (`
 `CF_Crc_t * crc,`
 `const uint8 * data,`
 `size_t len)`

Digest a chunk for CRC calculation.

Description

Does the CRC calculation, and stores an index into the given 4-byte word in case the input was not evenly divisible for 4.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<i>crc</i>	CRC object to operate on
<i>data</i>	Pointer to data to digest
<i>len</i>	Length of data to digest

Definition at line 53 of file cf_crc.c.

References CF_Crc::index, CF_Crc::result, and CF_Crc::working.

Referenced by CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_S_SendFileData().

12.41.2.2 CF_CRC_Finalize() `void CF_CRC_Finalize (`
 `CF_Crc_t * crc)`

Finalize a CRC calculation.

Description

Checks the index and if it isn't 0, does the final calculations on the bytes in the shift register. After this call is made, the result field of the structure holds the result.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<i>crc</i>	CRC object to operate on
------------	--------------------------

Definition at line 78 of file cf_crc.c.

References CF_Crc::index, CF_Crc::result, and CF_Crc::working.

Referenced by CF_CFDP_R_CheckCrc(), and CF_CFDP_S_SendEof().

12.41.2.3 CF_CRC_Start() `void CF_CRC_Start (`
 `CF_Crc_t * crc)`

Start a CRC streamable digest.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<code>crc</code>	CRC object to operate on
------------------	--------------------------

Definition at line 42 of file cf_crc.c.

Referenced by CF_CFDP_R2_CalcCrcChunk().

12.42 apps/cf/fsw/src(cf_crc.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct [CF_Crc](#)
CRC state object.

TypeDefs

- typedef struct [CF_Crc](#) [CF_Crc_t](#)
CRC state object.

Functions

- void [CF_CRC_Start](#) ([CF_Crc_t](#) *crc)
Start a CRC streamable digest.
- void [CF_CRC_Digest](#) ([CF_Crc_t](#) *crc, const [uint8](#) *data, [size_t](#) len)
Digest a chunk for CRC calculation.
- void [CF_CRC_Finalize](#) ([CF_Crc_t](#) *crc)
Finalize a CRC calculation.

12.42.1 Detailed Description

The CF Application CRC calculation header file

12.42.2 TypeDef Documentation

12.42.2.1 CF_Crc_t `typedef struct CF_Crc CF_Crc_t`
CRC state object.

12.42.3 Function Documentation

```
12.42.3.1 CF_CRC_Digest() void CF_CRC_Digest (
    CF_Crc_t * crc,
    const uint8 * data,
    size_t len )
```

Digest a chunk for CRC calculation.

Description

Does the CRC calculation, and stores an index into the given 4-byte word in case the input was not evenly divisible for 4.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<i>crc</i>	CRC object to operate on
<i>data</i>	Pointer to data to digest
<i>len</i>	Length of data to digest

Definition at line 53 of file cf_crc.c.

References CF_Crc::index, CF_Crc::result, and CF_Crc::working.

Referenced by CF_CFDP_R1_SubstateRecvFileData(), CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_S_SendFileData().

```
12.42.3.2 CF_CRC_Finalize() void CF_CRC_Finalize (
    CF_Crc_t * crc )
```

Finalize a CRC calculation.

Description

Checks the index and if it isn't 0, does the final calculations on the bytes in the shift register. After this call is made, the result field of the structure holds the result.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<i>crc</i>	CRC object to operate on
------------	--------------------------

Definition at line 78 of file cf_crc.c.

References CF_Crc::index, CF_Crc::result, and CF_Crc::working.

Referenced by CF_CFDP_R_CheckCrc(), and CF_CFDP_S_SendEof().

```
12.42.3.3 CF_CRC_Start() void CF_CRC_Start (
    CF_Crc_t * crc )
```

Start a CRC streamable digest.

Assumptions, External Events, and Notes:

crc must not be NULL.

Parameters

<i>crc</i>	CRC object to operate on
------------	--------------------------

Definition at line 42 of file cf_crc.c.

Referenced by CF_CFDP_R2_CalcCrcChunk().

12.43 apps/cf/fsw/src(cf_dispatch.c File Reference

```
#include "cf_dispatch.h"
#include "cf_app.h"
#include "cf_events.h"
#include "cf_cmd.h"
#include "cfe.h"
#include <string.h>
```

Functions

- void [CF_ProcessGroundCommand](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process any ground command contained in the given message.
- void [CF_AppPipe](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
CF message processing function.

12.43.1 Detailed Description

The CF Application main application source file

This file contains the functions that initialize the application and link all logic and functionality to the CFS.

12.43.2 Function Documentation

12.43.2.1 [CF_AppPipe\(\)](#) void CF_AppPipe (

```
    const CFE\_SB\_Buffer\_t * BufPtr )
```

CF message processing function.

Description

Process message packets received via the Software Bus command pipe

Assumptions, External Events, and Notes:

BufPtr must not be NULL.

Parameters

<i>in</i>	<i>BufPtr</i>	Software Bus message pointer
-----------	---------------	------------------------------

Definition at line 132 of file cf_dispatch.c.

Referenced by CF_AppMain().

```
12.43.2.2 CF_ProcessGroundCommand() void CF_ProcessGroundCommand (
    const CFE_SB_Buffer_t * BufPtr )
```

Process any ground command contained in the given message.

Assumptions, External Events, and Notes:

BuPtr must not be NULL.

Parameters

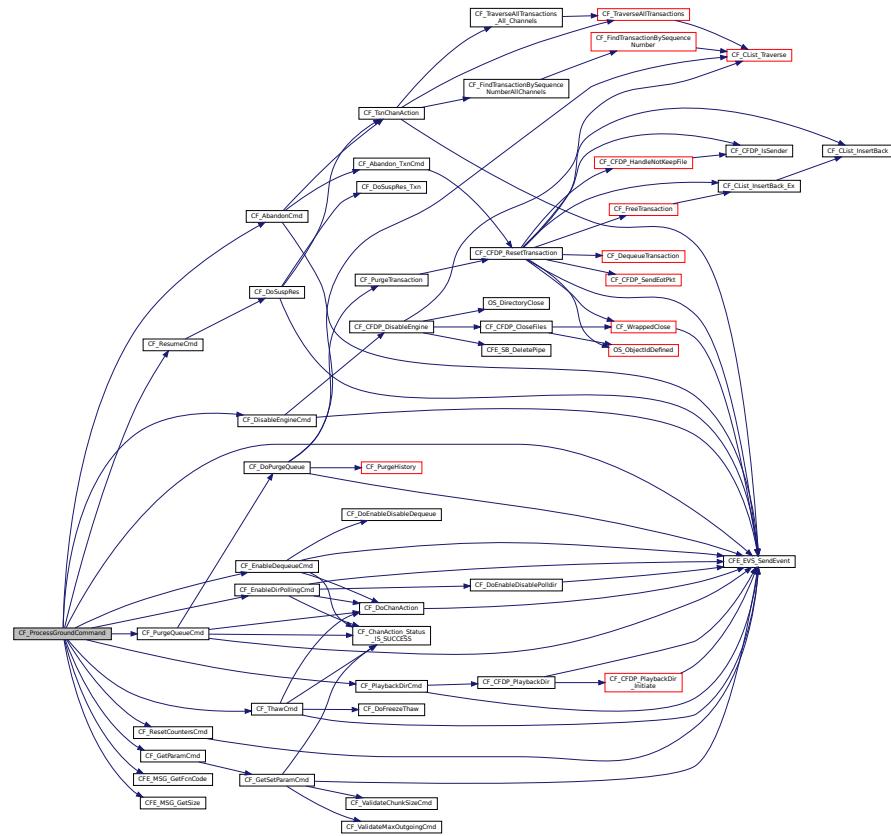
<i>BuPtr</i>	Pointer to command message
--------------	----------------------------

Definition at line 42 of file cf_dispatch.c.

References CF_ABANDON_CC, CF_AbandonCmd(), CF_AppData, CF_CANCEL_CC, CF_CC_ERR_EID, CF_CMD_LEN_ERR_EID, CF_DISABLE_DEQUEUE_CC, CF_DISABLE_DIR_POLLING_CC, CF_DISABLE_ENGINE_CC, CF_DisableEngineCmd(), CF_ENABLE_DEQUEUE_CC, CF_ENABLE_DIR_POLLING_CC, CF_ENABLE_ENGIN_CC, CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_FREEZE_CC, CF_GET_PARAM_CC, CF_GetParamCmd(), CF_NOOP_CC, CF_PLAYBACK_DIR_CC, CF_PlaybackDirCmd(), CF_PURGE_QUEUE_CC, CF_PurgeQueueCmd(), CF_RESET_CC, CF_ResetCountersCmd(), CF_RESUME_CC, CF_ResumeCmd(), CF_SET_PARAM_CC, CF_SUSPEND_CC, CF_THAW_CC, CF_ThawCmd(), CF_TX_FILE_CC, CF_WRITE_QUEUE_CC, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG_GetSize(), CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CFE_SB_Msg::Msg, and CF_HkPacket::Payload.

Referenced by CF_AppPipe().

Here is the call graph for this function:



12.44 apps/cf/fsw/src(cf_dispatch.h File Reference)

```
#include "cfe.h"
```

Functions

- void [CF_ProcessGroundCommand](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process any ground command contained in the given message.
 - void [CF_AppPipe](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
CF message processing function.

12.44.1 Detailed Description

The CF Application main application header file

12.44.2 Function Documentation

```
12.44.2.1 CF_AppPipe() void CF_AppPipe (           const CFE_SB_Buffer_t * BufPtr )
```

CF message processing function.

Description

Process message packets received via the Software Bus command pipe

Assumptions, External Events, and Notes:

BufPtr must not be NULL.

Parameters

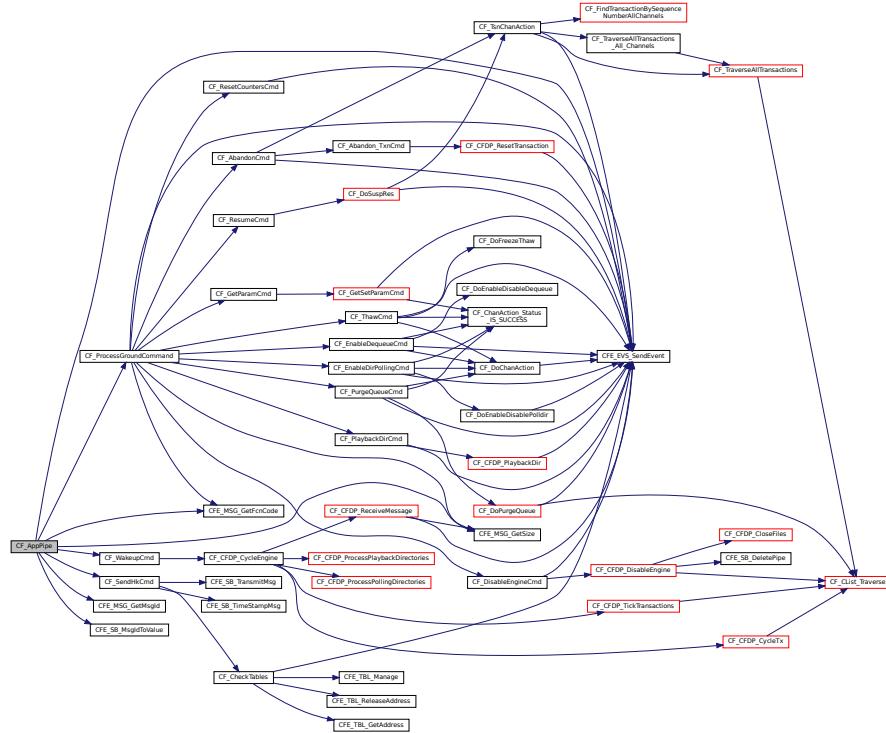
in *BufPtr* Software Bus message pointer

Definition at line 132 of file cf_dispatch.c.

References CF_AppData, CF_CC_ERR_EID, CF_CMD_LEN_ERR_EID, CF_CMD_MID, CF_MID_ERR_EID, CF_ProcessGroundCommand(), CF_SEND_HK_MID, CF_SendHkCmd(), CF_TC_DISPATCH_TABLE, CF_WAKE_UP_MID, CF_WakeupCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_E_MSG_GetMsgId(), CFE_MSG_GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), CFE_STATUS_UNKNOWN_MSG_ID, CFE_STATUS_WRONG_MSG_LENGTH, CFE_SUCCESS, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CFE_SB_Msg::Msg, and CF_HkPacket::Payload.

Referenced by CF_AppMain().

Here is the call graph for this function:



12.44.2.2 CF_ProcessGroundCommand() void CF_ProcessGroundCommand (

```
const CFE_SB_Buffer_t * BufPtr )
```

Process any ground command contained in the given message.

Assumptions, External Events, and Notes:

BufPtr must not be NULL.

Parameters

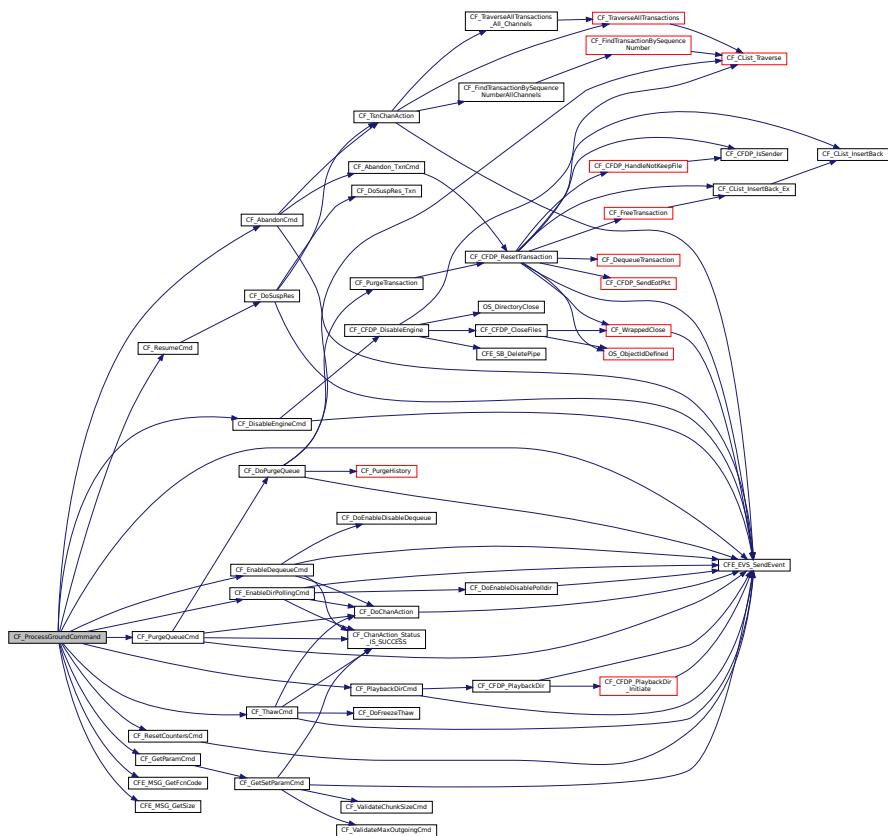
BufPtr Pointer to command message

Definition at line 42 of file cf_dispatch.c.

References CF_ABANDON_CC, CF_AbandonCmd(), CF_AppData, CF_CANCEL_CC, CF_CC_ERR_EID, CF_C←MD_LEN_ERR_EID, CF_DISABLE_DEQUEUE_CC, CF_DISABLE_DIR_POLLING_CC, CF_DISABLE_ENGINE_CC, CF_DisableEngineCmd(), CF_ENABLE_DEQUEUE_CC, CF_ENABLE_DIR_POLLING_CC, CF_ENABLE_ENGIN←E_CC, CF_EnableDequeueCmd(), CF_EnableDirPollingCmd(), CF_FREEZE_CC, CF_GET_PARAM_CC, CF_Get←ParamCmd(), CF_NOOP_CC, CF_PLAYBACK_DIR_CC, CF_PlaybackDirCmd(), CF_PURGE_QUEUE_CC, CF←PurgeQueueCmd(), CF_RESET_CC, CF_ResetCountersCmd(), CF_RESUME_CC, CF_ResumeCmd(), CF_SET←PARAM_CC, CF_SUSPEND_CC, CF_THAW_CC, CF_ThawCmd(), CF_TX_FILE_CC, CF_WRITE_QUEUE_CC, C←FE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG.GetSize(), CF_Hk←Packet_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CFE_SB_Msg::Msg, and CF_HkPacket::←Payload.

Referenced by CF_AppPipe().

Here is the call graph for this function:



12.45 apps/cf/fsw/src(cf_eds_dispatch.c File Reference)

```
#include "cf_app.h"
```

```
#include "cf_events.h"
#include "cf_dispatch.h"
#include "cf_cmd.h"
#include "cf_eds_dictionary.h"
#include "cf_eds_dispatcher.h"
#include "cfe_msg.h"
```

Functions

- void **CF_AppPipe** (const **CFE_SB_Buffer_t** *BufPtr)
CF message processing function.

Variables

- static const EdsDispatchTable_CF_Application_CFE_SB_Telecommand_t **CF_TC_DISPATCH_TABLE**

12.45.1 Function Documentation

12.45.1.1 CF_AppPipe() void CF_AppPipe (
 const **CFE_SB_Buffer_t** * *BufPtr*)
 CF message processing function.

Description

Process message packets received via the Software Bus command pipe

Assumptions, External Events, and Notes:

BufPtr must not be NULL.

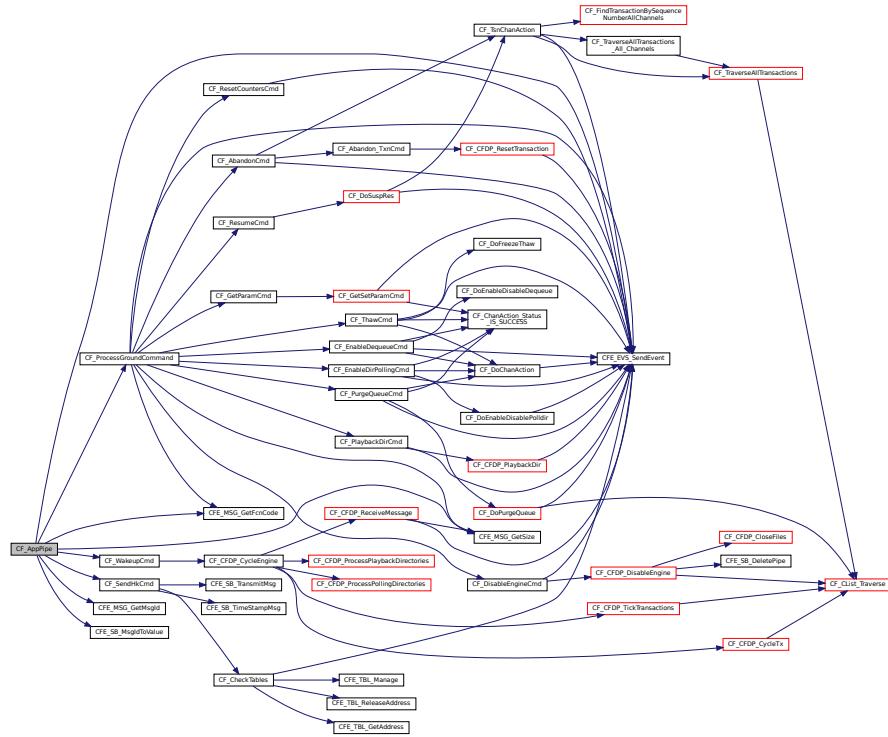
Parameters

in	<i>BufPtr</i>	Software Bus message pointer
----	---------------	------------------------------

Definition at line 44 of file cf_eds_dispatch.c.

References CF_AppData, CF_CC_ERR_EID, CF_CMD_LEN_ERR_EID, CF_CMD_MID, CF_MID_ERR_EID, CF_ProcessGroundCommand(), CF_SEND_HK_MID, CF_SendHkCmd(), CF_TC_DISPATCH_TABLE, CF_WAKE_UP_MID, CF_WakeupCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_E_MSG_GetMsgId(), CFE_MSG.GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), CFE_STATUS_UNKNOWN_MSG_ID, CFE_STATUS_WRONG_MSG_LENGTH, CFE_SUCCESS, CF_HkPacket_Payload::counters, CF_HkCmdCounters::err, CF_AppData_t::hk, CFE_SB_Msg::Msg, and CF_HkPacket::Payload.

Here is the call graph for this function:



12.45.2 Variable Documentation

12.45.2.1 CF_TC_DISPATCH_TABLE const EdsDispatchTable_CF_Application_CFE_SB_Telecommand_t CF↔

_TC_DISPATCH_TABLE [static]

Initial value:

```

= {
    .CMD =
    {
        .AbandonCmd_indication = CF_AbandonCmd,
        .CancelCmd_indication = CF_CancelCmd,
        .DisableDequeueCmd_indication = CF_DisableDequeueCmd,
        .DisableDirPollingCmd_indication = CF_DisableDirPollingCmd,
        .DisableEngineCmd_indication = CF_DisableEngineCmd,
        .EnableDequeueCmd_indication = CF_EnableDequeueCmd,
        .EnableDirPollingCmd_indication = CF_EnableDirPollingCmd,
        .EnableEngineCmd_indication = CF_EnableEngineCmd,
        .FreezeCmd_indication = CF_FreezeCmd,
        .GetParamCmd_indication = CF_GetParamCmd,
        .NoopCmd_indication = CF_NoopCmd,
        .PlaybackDirCmd_indication = CF_PlaybackDirCmd,
        .PurgeQueueCmd_indication = CF_PurgeQueueCmd,
        .ResetCountersCmd_indication = CF_ResetCountersCmd,
        .ResumeCmd_indication = CF_ResumeCmd,
        .SetParamCmd_indication = CF_SetParamCmd,
        .SuspendCmd_indication = CF_SuspendCmd,
        .ThawCmd_indication = CF_ThawCmd,
        .TxFileCmd_indication = CF_TxFileCmd,
        .WriteQueueCmd_indication = CF_WriteQueueCmd,
    },
    .SEND_HK = {.indication = CF_SendHkCmd},
    .WAKE_UP = {.indication = CF_WakeupCmd}}
}

```

Definition at line 11 of file cf_eds_dispatch.c.

Referenced by CF_AppPipe().

12.46 apps/cf/fsw/src(cf_logical_pdu.h File Reference

```
#include "common_types.h"
#include "cf_extern_typedefs.h"
#include "cf_cfdp_pdu.h"
```

Data Structures

- struct [CF_Logical_PduHeader](#)
Structure representing base CFDP PDU header.
- struct [CF_Logical_PduFileDirectiveHeader](#)
Structure representing logical File Directive header.
- struct [CF_Logical_Lv](#)
Structure representing logical LV Object format.
- union [CF_Logical_TlvData](#)
Union of various data items that may occur in a TLV item.
- struct [CF_Logical_Tlv](#)
Structure representing logical TLV Object format.
- struct [CF_Logical_SegmentRequest](#)
Structure representing logical Segment Request data.
- struct [CF_Logical_SegmentList](#)
- struct [CF_Logical_TlvList](#)
- struct [CF_Logical_PduEof](#)
Structure representing logical End of file PDU.
- struct [CF_Logical_PduFin](#)
Structure representing logical Finished PDU.
- struct [CF_Logical_PduAck](#)
Structure representing CFDP Acknowledge PDU.
- struct [CF_Logical_PduMd](#)
Structure representing CFDP Metadata PDU.
- struct [CF_Logical_PduNak](#)
Structure representing logical Non-Acknowledge PDU.
- struct [CF_Logical_PduFileDataHeader](#)
- union [CF_Logical_IntHeader](#)
A union of all possible internal header types in a PDU.
- struct [CF_Logical_PduBuffer](#)
Encapsulates the entire PDU information.

Macros

- #define [CF_PDU_MAX_TLV](#) (4)
Maximum number of TLV values in a single PDU.
- #define [CF_PDU_MAX_SEGMENTS](#) ([CF_NAK_MAX_SEGMENTS](#))
Maximum number of segment requests in a single PDU.

Typedefs

- **typedef uint32 CF_FileSize_t**
Type for logical file size/offset value.
- **typedef struct CF_Logical_PduHeader CF_Logical_PduHeader_t**
Structure representing base CFDP PDU header.
- **typedef struct CF_Logical_PduFileDirectiveHeader CF_Logical_PduFileDirectiveHeader_t**
Structure representing logical File Directive header.
- **typedef struct CF_Logical_Lv CF_Logical_Lv_t**
Structure representing logical LV Object format.
- **typedef union CF_Logical_TlvData CF_Logical_TlvData_t**
Union of various data items that may occur in a TLV item.
- **typedef struct CF_Logical_Tlv CF_Logical_Tlv_t**
Structure representing logical TLV Object format.
- **typedef struct CF_Logical_SegmentRequest CF_Logical_SegmentRequest_t**
Structure representing logical Segment Request data.
- **typedef struct CF_Logical_SegmentList CF_Logical_SegmentList_t**
- **typedef struct CF_Logical_TlvList CF_Logical_TlvList_t**
- **typedef struct CF_Logical_PduEof CF_Logical_PduEof_t**
Structure representing logical End of file PDU.
- **typedef struct CF_Logical_PduFin CF_Logical_PduFin_t**
Structure representing logical Finished PDU.
- **typedef struct CF_Logical_PduAck CF_Logical_PduAck_t**
Structure representing CFDP Acknowledge PDU.
- **typedef struct CF_Logical_PduMd CF_Logical_PduMd_t**
Structure representing CFDP Metadata PDU.
- **typedef struct CF_Logical_PduNak CF_Logical_PduNak_t**
Structure representing logical Non-Acknowledge PDU.
- **typedef struct CF_Logical_PduFileHeader CF_Logical_PduFileHeader_t**
- **typedef union CF_Logical_IntHeader CF_Logical_IntHeader_t**
A union of all possible internal header types in a PDU.
- **typedef struct CF_Logical_PduBuffer CF_Logical_PduBuffer_t**
Encapsulates the entire PDU information.

12.46.1 Detailed Description

Structures defining logical CFDP PDUs

These are CF-specific data structures that reflect the logical content of the CFDP PDUs defined in [cf_cfdp_pdu.h](#). Note these are *NOT* intended to reflect the bitwise structures defined in the CCSDS blue book, but rather the values contained within those structures, in a form that can be used by software.

Specifically, this intent differs in the following ways:

- All numeric fields are in native byte order
- All structures are padded/aligned according to native CPU (i.e. not packed)
- All bit-fields are exploded, where each field/group is a separate member
- Variable-size content is normalized, allocated as the maximum possible size

12.46.2 Macro Definition Documentation

12.46.2.1 CF_PDU_MAX_SEGMENTS #define CF_PDU_MAX_SEGMENTS (CF_NAK_MAX_SEGMENTS)

Maximum number of segment requests in a single PDU.

Sets an upper bound on the logical structures for the most possible segment structures in a single PDU.

Definition at line 66 of file cf_logical_pdu.h.

12.46.2.2 CF_PDU_MAX_TLV #define CF_PDU_MAX_TLV (4)

Maximum number of TLV values in a single PDU.

This just serves to set an upper bound on the logical structures, to keep things simple. The real limit varies depending on the specific PDU type being processed. This caps the amount of storage memory for the worst case, the actual number present is always part of the run-time state.

Without filestore requests, use of TLV is pretty limited.

Definition at line 58 of file cf_logical_pdu.h.

12.46.3 Typedef Documentation

12.46.3.1 CF_FileSize_t typedef uint32 CF_FileSize_t

Type for logical file size/offset value.

The CFDP protocol permits use of 64-bit values for file size/offsets Although the CF application only supports 32-bit legacy file size type at this point, the logical structures should use this type in case future support for large files is added.

Definition at line 76 of file cf_logical_pdu.h.

12.46.3.2 CF_Logical_IntHeader_t typedef union CF_Logical_IntHeader CF_Logical_IntHeader_t

A union of all possible internal header types in a PDU.

The specific entry which applies depends on the combination of pdu type and directive code.

12.46.3.3 CF_Logical_Lv_t typedef struct CF_Logical_Lv CF_Logical_Lv_t

Structure representing logical LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

These are only used for string data (mostly filenames) so the data can refer directly to the encoded bits, it does not necessarily need to be duplicated here.

12.46.3.4 CF_Logical_PduAck_t typedef struct CF_Logical_PduAck CF_Logical_PduAck_t

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

12.46.3.5 CF_Logical_PduBuffer_t typedef struct CF_Logical_PduBuffer CF_Logical_PduBuffer_t

Encapsulates the entire PDU information.

12.46.3.6 CF_Logical_PduEof_t typedef struct CF_Logical_PduEof CF_Logical_PduEof_t

Structure representing logical End of file PDU.

See also

[CF_CFDP_PduEof_t](#) for encoded form

12.46.3.7 CF_Logical_PduFileDataHeader_t typedef struct CF_Logical_PduFileDataHeader CF_Logical_PduFileDataHeader_t

12.46.3.8 CF_Logical_PduFileDirectiveHeader_t `typedef struct CF_Logical_PduFileDirectiveHeader CF_Logical_PduFileDirectiveHeader_t`

Structure representing logical File Directive header.

This contains the file directive code from the PDUs for which it applies. The codes are mapped directly to the CFDP protocol values, but converted to a native value (enum) for direct use by software.

12.46.3.9 CF_Logical_PduFin_t `typedef struct CF_Logical_PduFin CF_Logical_PduFin_t`

Structure representing logical Finished PDU.

See also

[CF_CFDP_PduFin_t](#) for encoded form

12.46.3.10 CF_Logical_PduHeader_t `typedef struct CF_Logical_PduHeader CF_Logical_PduHeader_t`

Structure representing base CFDP PDU header.

Reflects the common content at the beginning of all CFDP PDUs, of all types.

See also

[CF_CFDP_PduHeader_t](#) for encoded form

12.46.3.11 CF_Logical_PduMd_t `typedef struct CF_Logical_PduMd CF_Logical_PduMd_t`

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

12.46.3.12 CF_Logical_PduNak_t `typedef struct CF_Logical_PduNak CF_Logical_PduNak_t`

Structure representing logical Non-Acknowledge PDU.

12.46.3.13 CF_Logical_SegmentList_t `typedef struct CF_Logical_SegmentList CF_Logical_SegmentList_t`**12.46.3.14 CF_Logical_SegmentRequest_t** `typedef struct CF_Logical_SegmentRequest CF_Logical_SegmentRequest_t`
Structure representing logical Segment Request data.**12.46.3.15 CF_Logical_Tlv_t** `typedef struct CF_Logical_Tlv CF_Logical_Tlv_t`

Structure representing logical TLV Object format.

In the current implementation of CF, only entity IDs are currently encoded in this form where indicated in the spec. This may change in a future version.

See also

[CF_CFDP_tlv_t](#) for encoded form

12.46.3.16 CF_Logical_TlvData_t `typedef union CF_Logical_TlvData CF_Logical_TlvData_t`

Union of various data items that may occur in a TLV item.

The actual type is identified by the "type" field in the enclosing TLV

Currently filestore requests are not implemented in CF, so the TLV use is limited. This may change in the future.

Numeric data needs to actually be copied to this buffer, because it needs to be normalized in length and byte-order. But string data (e.g. filenames, messages) can reside in the original encoded form.

12.46.3.17 CF_Logical_TlvList_t `typedef struct CF_Logical_TlvList CF_Logical_TlvList_t`

12.47 apps/cf/fsw/src(cf_timer.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_timer.h"
#include "cf_app.h"
#include "cf_assert.h"
```

Functions

- `uint32 CF_Timer_Sec2Ticks (CF_Timer_Seconds_t sec)`
Converts seconds into scheduler ticks.
- `void CF_Timer_InitRelSec (CF_Timer_t *txn, uint32 rel_sec)`
Initialize a timer with a relative number of seconds.
- `bool CF_Timer_Expired (const CF_Timer_t *txn)`
Check if a timer has expired.
- `void CF_Timer_Tick (CF_Timer_t *txn)`
Notify a timer object a tick has occurred.

12.47.1 Detailed Description

The CF Application timer source file

A timer in CF is really just a structure that holds a counter that indicates the timer expired when it reaches 0. The goal is that any timer is driven by the scheduler ticks. There is no reason we need any finer grained resolution than this for CF.

12.47.2 Function Documentation

12.47.2.1 CF_Timer_Expired() `bool CF_Timer_Expired (`

`const CF_Timer_t * txn)`

Check if a timer has expired.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<code>txn</code>	Timer object to check
------------------	-----------------------

Returns

status code indicating whether timer has expired

Return values

<code>1</code>	if expired
<code>0</code>	if not expired

Definition at line 65 of file cf_timer.c.

References CF_Timer::tick.

Referenced by CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

12.47.2.2 CF_Timer_InitRelSec() `void CF_Timer_InitRelSec (`
`CF_Timer_t * txn,`
`CF_Timer_Seconds_t rel_sec)`

Initialize a timer with a relative number of seconds.

Assumptions, External Events, and Notes:

`txn` must not be NULL.

Parameters

<code>txn</code>	Timer object to initialize
<code>rel_sec</code>	Relative number of seconds

Definition at line 54 of file cf_timer.c.

References CF_Timer_Sec2Ticks(), and CF_Timer::tick.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), and CF_CFDP_ProcessPollingDirectories().

Here is the call graph for this function:



12.47.2.3 CF_Timer_Sec2Ticks() `uint32 CF_Timer_Sec2Ticks (`
`CF_Timer_Seconds_t sec)`

Converts seconds into scheduler ticks.

Assumptions, External Events, and Notes:

sub-second resolution is not required

Parameters

<code>sec</code>	Number of seconds
------------------	-------------------

Returns

Number of ticks for the given seconds.

Definition at line 43 of file cf_timer.c.

References CF_AppData, CF_AppData_t::config_table, and CF_ConfigTable::ticks_per_second.
Referenced by CF_Timer_InitRelSec().

12.47.2.4 CF_Timer_Tick()

```
void CF_Timer_Tick (
    CF_Timer_t * txn )
```

Notify a timer object a tick has occurred.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Timer object to tick
------------	----------------------

Definition at line 76 of file cf_timer.c.

References CF_ASSERT, and CF_Timer::tick.

Referenced by CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

12.48 apps/cf/fsw/src(cf_timer.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct [CF_Timer](#)
Basic CF timer object.

Typedefs

- [typedef uint32 CF_Timer_Ticks_t](#)
Type for a timer tick count.
- [typedef uint32 CF_Timer_Seconds_t](#)
Type for a timer number of seconds.
- [typedef struct CF_Timer CF_Timer_t](#)
Basic CF timer object.

Functions

- [void CF_Timer_InitRelSec \(CF_Timer_t *txn, CF_Timer_Seconds_t rel_sec\)](#)
Initialize a timer with a relative number of seconds.
- [bool CF_Timer_Expired \(const CF_Timer_t *txn\)](#)
Check if a timer has expired.
- [void CF_Timer_Tick \(CF_Timer_t *txn\)](#)
Notify a timer object a tick has occurred.
- [uint32 CF_Timer_Sec2Ticks \(CF_Timer_Seconds_t sec\)](#)
Converts seconds into scheduler ticks.

12.48.1 Detailed Description

The CF Application timer header file

12.48.2 Typedef Documentation

12.48.2.1 CF_Timer_Seconds_t `typedef uint32 CF_Timer_Seconds_t`

Type for a timer number of seconds.

Definition at line 43 of file cf_timer.h.

12.48.2.2 CF_Timer_t `typedef struct CF_Timer CF_Timer_t`

Basic CF timer object.

12.48.2.3 CF_Timer_Ticks_t `typedef uint32 CF_Timer_Ticks_t`

Type for a timer tick count.

Note

We expect ticks to be 100/sec, so using uint32 for sec could have a bounds condition with uint32. But, we don't expect to use more than 400,000,000 seconds for any reason so let's just live with it.

Definition at line 38 of file cf_timer.h.

12.48.3 Function Documentation

12.48.3.1 CF_Timer_Expired() `bool CF_Timer_Expired (` `const CF_Timer_t * txn)`

Check if a timer has expired.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<code>txn</code>	Timer object to check
------------------	-----------------------

Returns

status code indicating whether timer has expired

Return values

<code>1</code>	if expired
<code>0</code>	if not expired

Definition at line 65 of file cf_timer.c.

References CF_Timer::tick.

Referenced by CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

12.48.3.2 CF_Timer_InitRelSec() `void CF_Timer_InitRelSec (CF_Timer_t * txn,
CF_Timer_Seconds_t rel_sec)`

Initialize a timer with a relative number of seconds.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Timer object to initialize
<i>rel_sec</i>	Relative number of seconds

Definition at line 54 of file cf_timer.c.

References CF_Timer_Sec2Ticks(), and CF_Timer::tick.

Referenced by CF_CFDP_ArmAckTimer(), CF_CFDP_ArmInactTimer(), and CF_CFDP_ProcessPollingDirectories().

Here is the call graph for this function:



12.48.3.3 CF_Timer_Sec2Ticks() `uint32 CF_Timer_Sec2Ticks (CF_Timer_Seconds_t sec)`

Converts seconds into scheduler ticks.

Assumptions, External Events, and Notes:

sub-second resolution is not required

Parameters

<i>sec</i>	Number of seconds
------------	-------------------

Returns

Number of ticks for the given seconds.

Definition at line 43 of file cf_timer.c.

References CF_AppData, CF_AppData_t::config_table, and CF_ConfigTable::ticks_per_second.

Referenced by CF_Timer_InitRelSec().

12.48.3.4 CF_Timer_Tick() `void CF_Timer_Tick (CF_Timer_t * txn)`

Notify a timer object a tick has occurred.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

<i>txn</i>	Timer object to tick
------------	----------------------

Definition at line 76 of file cf_timer.c.

References CF_ASSERT, and CF_Timer:::tick.

Referenced by CF_CFDP_ProcessPollingDirectories(), CF_CFDP_R_Tick(), and CF_CFDP_S_Tick().

12.49 apps/cf/fsw/src(cf_utils.c File Reference

```
#include "cf_app.h"
#include "cf_verify.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_events.h"
#include "cf_perfids.h"
#include "cf_assert.h"
```

Functions

- [CF_Transaction_t * CF_FindUnusedTransaction \(CF_Channel_t *chan\)](#)
Find an unused transaction on a channel.
- [void CF_ResetHistory \(CF_Channel_t *chan, CF_History_t *history\)](#)
Returns a history structure back to its unused state.
- [void CF_FreeTransaction \(CF_Transaction_t *txn\)](#)
Frees and resets a transaction and returns it for later use.
- [CFE_Status_t CF_FindTransactionBySequenceNumber_Impl \(CF_CListNode_t *node, CF_Traverse_TransSeqArg_t *context\)](#)
List traversal function to check if the desired sequence number matches.
- [CF_Transaction_t * CF_FindTransactionBySequenceNumber \(CF_Channel_t *chan, CF_TransactionSeq_t transaction_sequence_number, CF_EntityId_t src_eid\)](#)
Finds an active transaction by sequence number.
- [CFE_Status_t CF_WriteHistoryEntryToFile \(osal_id_t fd, const CF_History_t *history\)](#)
Write a single history to a file.
- [CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile \(CF_CListNode_t *node, void *arg\)](#)
Writes a human readable representation of a history queue entry to a file.
- [CF_CListTraverse_Status_t CF_Traverse_WriteTxnQueueEntryToFile \(CF_CListNode_t *node, void *arg\)](#)
Writes a human readable representation of a transaction history entry to a file.
- [CFE_Status_t CF_WriteTxnQueueDataToFile \(osal_id_t fd, CF_Channel_t *chan, CF_QueueIdx_t queue\)](#)
Write a transaction-based queue's transaction history to a file.
- [CFE_Status_t CF_WriteHistoryQueueDataToFile \(osal_id_t fd, CF_Channel_t *chan, CF_Direction_t dir\)](#)
Write a history-based queue's entries to a file.
- [CF_CListTraverse_Status_t CF_PrioSearch \(CF_CListNode_t *node, void *context\)](#)
Searches for the first transaction with a lower priority than given.

- void `CF_InsertSortPrio (CF_Transaction_t *txn, CF_QueueIdx_t queue)`
Insert a transaction into a priority sorted transaction queue.
- `CF_CListTraverse_Status_t CF_TraverseAllTransactions_Impl (CF_CListNode_t *node, void *arg)`
List traversal function performs operation on every active transaction.
- `int32 CF_TraverseAllTransactions (CF_Channel_t *chan, CF_TraverseAllTransactions_fn_t fn, void *context)`
Traverses all transactions on all active queues and performs an operation on them.
- `int32 CF_TraverseAllTransactions_All_Channels (CF_TraverseAllTransactions_fn_t fn, void *context)`
Traverses all transactions on all channels and performs an operation on them.
- `CFE_Status_t CF_WrappedOpenCreate (osal_id_t *fd, const char *fname, int32 flags, int32 access)`
Wrap the filesystem open call with a perf counter.
- void `CF_WrappedClose (osal_id_t fd)`
Wrap the filesystem close call with a perf counter.
- `CFE_Status_t CF_WrappedRead (osal_id_t fd, void *buf, size_t read_size)`
Wrap the filesystem read call with a perf counter.
- `CFE_Status_t CF_WrappedWrite (osal_id_t fd, const void *buf, size_t write_size)`
Wrap the filesystem write call with a perf counter.
- `CFE_Status_t CF_WrappedLseek (osal_id_t fd, off_t offset, int mode)`
Wrap the filesystem lseek call with a perf counter.
- bool `CF_TxnStatus_IsError (CF_TxnStatus_t txn_stat)`
Check if the internal transaction status represents an error.
- `CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode (CF_TxnStatus_t txn_stat)`
Converts the internal transaction status to a CFDP condition code.
- `CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (CF_CFDP_ConditionCode_t cc)`
Converts a CFDP condition code to an internal transaction status.

12.49.1 Detailed Description

The CF Application general utility functions source file
 Various odds and ends are put here.

12.49.2 Function Documentation

```
12.49.2.1 CF_FindTransactionBySequenceNumber() CF_Transaction_t* CF_FindTransactionBySequence←
Number (
    CF_Channel_t * chan,
    CF_TransactionSeq_t transaction_sequence_number,
    CF_EntityId_t src_eid )
```

Finds an active transaction by sequence number.

Description

This function traverses the active rx, pending, txa, and txw transaction and looks for the requested transaction.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>chan</i>	Pointer to the CF channel
<i>transaction_sequence_number</i>	Sequence number to find
<i>src_eid</i>	Entity ID associated with sequence number

Returns

Pointer to the given transaction if found

Return values

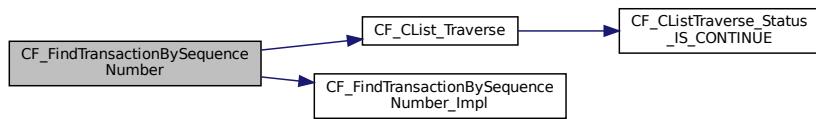
<i>NULL</i>	if the transaction is not found
-------------	---------------------------------

Definition at line 139 of file cf_utils.c.

References CF_CList_Traverse(), CF_FindTransactionBySequenceNumber_Impl(), CF_QueueIdx_PEND, CF_QueueIdx_RX, CF_QueueIdx_TXA, CF_QueueIdx_TXW, CF_Channel::qs, and CF_Traverse_TransSeqArg::txn.

Referenced by CF_CFDP_ReceiveMessage(), and CF_FindTransactionBySequenceNumberAllChannels().

Here is the call graph for this function:



```

12.49.2.2 CF_FindTransactionBySequenceNumber_Impl() CFE_Status_t CF_FindTransactionBySequenceNumber_Impl (
    CF_CListNode_t * node,
    CF_Traverse_TransSeqArg_t * context )
  
```

List traversal function to check if the desired sequence number matches.

Assumptions, External Events, and Notes:

context must not be NULL. node must not be NULL.

Parameters

<i>node</i>	Pointer to node currently being traversed
<i>context</i>	Pointer to state object passed through from initial call

Return values

<i>1</i>	when it's found, which terminates list traversal
<i>0</i>	when it isn't found, which causes list traversal to continue

Definition at line 119 of file cf_utils.c.

References container_of, CF_Transaction::history, CF_History::seq_num, CF_Traverse_TransSeqArg::src_eid, CF_<History::src_eid, CF_Traverse_TransSeqArg::transaction_sequence_number, and CF_Traverse_TransSeqArg::txn. Referenced by CF_FindTransactionBySequenceNumber().

12.49.2.3 CF_FindUnusedTransaction()

```
CF_Transaction_t* CF_FindUnusedTransaction (
    CF_Channel_t * chan )
```

Find an unused transaction on a channel.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

chan	Pointer to the CF channel
------	---------------------------

Returns

Pointer to a free transaction

Return values

NULL	if no free transactions available.
------	------------------------------------

Definition at line 43 of file cf_utils.c.

References CF_ASSERT, CF_CList_Remove_Ex(), CF_Direction_NUM, CF_QueueIdx_FREE, CF_QueueIdx_HIST, CF_QueueIdx_HIST_FREE, CF_History::cl_node, CF_Transaction::cl_node, container_of, CF_History::dir, CF_<Transaction::history, and CF_Channel::qs. Referenced by CF_CFDP_ProcessPlaybackDirectory(), CF_CFDP_ReceiveMessage(), and CF_CFDP_TxFile().

Here is the call graph for this function:



12.49.2.4 CF_FreeTransaction()

```
void CF_FreeTransaction (
    CF_Transaction_t * txn )
```

Frees and resets a transaction and returns it for later use.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

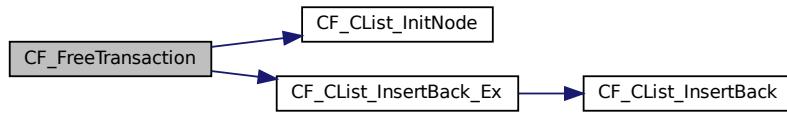
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 101 of file cf_utils.c.

References CF_AppData, CF_CList_InitNode(), CF_CList_InsertBack_Ex(), CF_QueueIdx_FREE, CF_TxnState_IDLE, CF_Transaction::chan_num, CF_Engine::channels, CF_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::engine, CF_Transaction::fd, CF_Transaction::flags, OS_OBJECT_ID_UNDEFINED, CF_Flags_Common::q_index, and CF_Transaction::state.

Referenced by CF_CFDP_InitEngine(), and CF_CFDP_ResetTransaction().

Here is the call graph for this function:



12.49.2.5 CF_InsertSortPrio() void CF_InsertSortPrio (

```

    CF_Transaction_t * txn,
    CF_QueueIdx_t queue )

```

Insert a transaction into a priority sorted transaction queue.

Description

This function works by walking the queue in reverse to find a transaction with a higher priority than the given transaction. The given transaction is then inserted after that one, since it would be the next lower priority.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

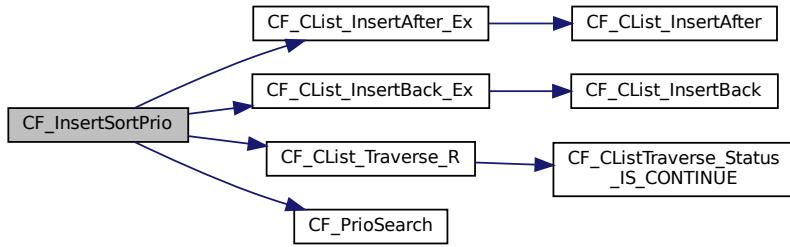
<i>txn</i>	Pointer to the transaction object
<i>queue</i>	Index of queue to insert into

Definition at line 332 of file cf_utils.c.

References CF_AppData, CF_Assert, CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Traverse_R(), CF_NUM_CHANNELS, CF_PrioSearch(), CF_TxnState_IDLE, CF_Transaction::chan_num, CF_Engine::channels, CF_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::engine, CF_Transaction::flags, CF_Transaction::priority, CF_Flags_Common::q_index, CF_Channel::qs, CF_Transaction::state, and CF_Traverse_PriorityArg::txn.

Referenced by CF_CFDP_S2_SubstateSendEof(), and CF_CFDP_TxFile_Initiate().

Here is the call graph for this function:



12.49.2.6 CF_PrioSearch() `CF_CListTraverse_Status_t CF_PrioSearch (`
`CF_CListNode_t * node,`
`void * context)`

Searches for the first transaction with a lower priority than given.

Assumptions, External Events, and Notes:

node must not be NULL. context must not be NULL.

Parameters

<code>node</code>	Node being currently traversed
<code>context</code>	Pointer to <code>CF_Traverse_PriorityArg_t</code> object indicating the priority to search for

Return values

<code>CF_CLIST_EXIT</code>	when it's found, which terminates list traversal
<code>CF_CLIST_CONT</code>	when it isn't found, which causes list traversal to continue

Definition at line 308 of file cf_utils.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `container_of`, `CF_Traverse_PriorityArg::priority`, `CF_Transaction::priority`, and `CF_Traverse_PriorityArg::txn`.

Referenced by `CF_InsertSortPrio()`.

12.49.2.7 CF_ResetHistory() `void CF_ResetHistory (`
`CF_Channel_t * chan,`
`CF_History_t * history)`

Returns a history structure back to its unused state.

Description

There's nothing to do currently other than remove the history from its current queue and put it back on `CF_QueueIdx_HIST_FREE`.

Assumptions, External Events, and Notes:

chan must not be NULL. history must not be NULL.

Parameters

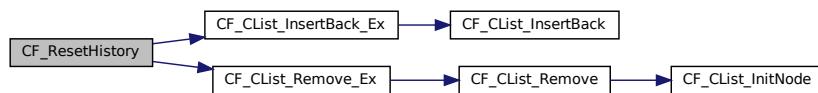
<i>chan</i>	Pointer to the CF channel
<i>history</i>	Pointer to the history entry

Definition at line 89 of file cf_utils.c.

References CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_QueueIdx_HIST, CF_QueueIdx_HIST_FREE, and CF_History::cl_node.

Referenced by CF_PurgeHistory().

Here is the call graph for this function:

**12.49.2.8 CF_Traverse_WriteHistoryQueueEntryToFile()** *CF_CListTraverse_Status_t* CF_Traverse_Write←

```
HistoryQueueEntryToFile (
    CF_CListNode_t * node,
    void * arg )
```

Writes a human readable representation of a history queue entry to a file.

This function is a wrapper around [CF_WriteHistoryEntryToFile\(\)](#) that can be used with CF_Traverse() to write history queue entries to the file.

This also implements a direction filter, so only matching entries are actually written to the file.

Assumptions, External Events, and Notes:

node must not be NULL. arg must not be NULL.

Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to CF_Traverse_WriteHistoryFileArg_t indicating the file information

Return values

<i>CF_CLIST_CONT</i>	if everything is going well
<i>CF_CLIST_EXIT</i>	if a write error occurred, which means traversal should stop

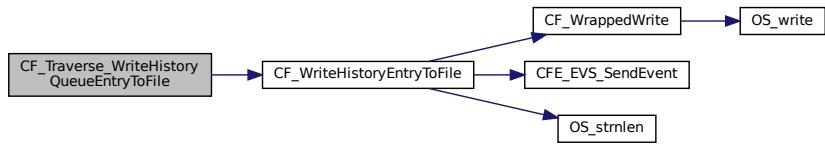
Definition at line 222 of file cf_utils.c.

References CF_CLIST_CONT, CF_CLIST_EXIT, CF_Direction_NUM, CF_WriteHistoryEntryToFile(), container_of, CF_Traverse_WriteHistoryFileArg::counter, CF_History::dir, CF_Traverse_WriteHistoryFileArg::error, CF_Traverse←

WriteHistoryFileArg::fd, and CF_Traverse_WriteHistoryFileArg::filter_dir.

Referenced by CF_WriteHistoryQueueDataToFile().

Here is the call graph for this function:



12.49.2.9 CF_Traverse_WriteTxnQueueEntryToFile()

```
CF_CListTraverse_Status_t CF_Traverse_WriteTxnQueueEntryToFile (
    CF_CListNode_t * node,
    void * arg )
```

Writes a human readable representation of a transaction history entry to a file.

This function is a wrapper around `CF_WriteHistoryEntryToFile()` that can be used with `CF_Traverse()` to write transaction queue entries to the file.

Assumptions, External Events, and Notes:

`node` must not be NULL. `arg` must not be NULL.

Parameters

<code>node</code>	Node being currently traversed
<code>arg</code>	Pointer to <code>CF_Traverse_WriteTxnFileArg_t</code> indicating the file information

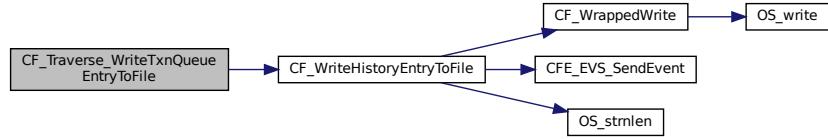
Return values

<code>CF_CLIST_CONT</code>	if everything is going well
<code>CF_CLIST_EXIT</code>	if a write error occurred, which means traversal should stop

Definition at line 249 of file cf_utils.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `CF_WriteHistoryEntryToFile()`, `container_of`, `CF_Traverse_WriteTxnToFileArg::counter`, `CF_Traverse_WriteTxnFileArg::error`, `CF_Traverse_WriteTxnFileArg::fd`, and `CF_Transaction::history`. Referenced by `CF_WriteTxnQueueDataToFile()`.

Here is the call graph for this function:



12.49.2.10 CF_TraverseAllTransactions() `int32 CF_TraverseAllTransactions (CF_Channel_t * chan, CF_TraverseAllTransactions_fn_t fn, void * context)`

Traverses all transactions on all active queues and performs an operation on them.

Assumptions, External Events, and Notes:

chan must not be NULL. fn must be a valid function. context must not be NULL.

Parameters

<i>chan</i>	Channel to operate on
<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

Returns

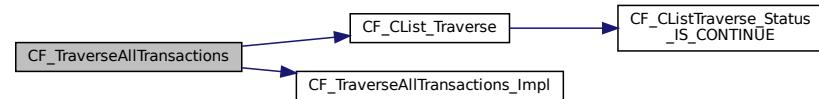
Number of transactions traversed

Definition at line 390 of file cf_utils.c.

References CF_CList_Traverse(), CF_QueueIdx_PEND, CF_QueueIdx_RX, CF_TraverseAllTransactions_Impl(), CF_TraverseAll_Arg::counter, and CF_Channel::qs.

Referenced by CF_TraverseAllTransactions_All_Channels(), and CF_TsnChanAction().

Here is the call graph for this function:



12.49.2.11 CF_TraverseAllTransactions_All_Channels() `int32 CF_TraverseAllTransactions_All_Channels (`

```
CF_TraverseAllTransactions_fn_t fn,
void * context )
```

Traverses all transactions on all channels and performs an operation on them.

Assumptions, External Events, and Notes:

fn must be a valid function. context must not be NULL.

Parameters

<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

Returns

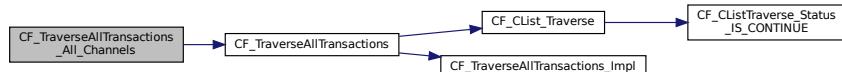
Number of transactions traversed

Definition at line 406 of file cf_utils.c.

References CF_AppData, CF_NUM_CHANNELS, CF_TraverseAllTransactions(), CF_Engine::channels, and CF_AppData_t::engine.

Referenced by CF_TsnChanAction().

Here is the call graph for this function:



12.49.2.12 CF_TraverseAllTransactions_Impl() CF_CListTraverse_Status_t CF_TraverseAllTransactions_Impl(

```
CF_CListNode_t * node,
void * arg )
```

List traversal function performs operation on every active transaction.

Description

Called on every transaction via list traversal. Calls another function on that transaction.

Assumptions, External Events, and Notes:

node must not be NULL. args must not be NULL.

Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Intermediate context object from initial call

Return values

<i>0</i>	for do not exit early (always continue)
----------	---

Definition at line 375 of file cf_utils.c.

References CF_CLIST_CONT, container_of, CF_TraverseAll_Arg::context, CF_TraverseAll_Arg::counter, and CF_TraverseAll_Arg::fn.

Referenced by CF_TraverseAllTransactions().

12.49.2.13 CF_TxnStatus_From_ConditionCode() `CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (CF_CFDP_ConditionCode_t cc)`

Converts a CFDP condition code to an internal transaction status.

Assumptions, External Events, and Notes:

None

Parameters

<i>cc</i>	CFDP condition code
-----------	---------------------

Returns

Transaction status code

Definition at line 589 of file cf_utils.c.

Referenced by CF_CFDP_R2_SubstateRecvEof().

12.49.2.14 CF_TxnStatus_IsError() `bool CF_TxnStatus_IsError (CF_TxnStatus_t txn_stat)`

Check if the internal transaction status represents an error.

Assumptions, External Events, and Notes:

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

Parameters

<i>txn_stat</i>	Transaction status
-----------------	--------------------

Returns

Boolean value indicating if the transaction is in an errored state

Return values

<i>true</i>	if an error has occurred during the transaction
<i>false</i>	if no error has occurred during the transaction yet

Definition at line 507 of file cf_utils.c.

References CF_TxnStatus_NO_ERROR.

Referenced by CF_CFDP_HandleNotKeepFile(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_DispatchRecv(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_SetTxnStatus(), and CF_TxnStatus_To_ConditionCode().

12.49.2.15 CF_TxnStatus_To_ConditionCode() CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode

```
(  
    CF_TxnStatus_t txn_stat )
```

Converts the internal transaction status to a CFDP condition code.

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

Assumptions, External Events, and Notes:

Not all transaction status codes directly correlate to a CFDP CC

Parameters

txn_stat	Transaction status
----------	--------------------

Returns

CFDP protocol condition code

Definition at line 523 of file cf_utils.c.

References CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED, CF_CFDP_ConditionCode_INACTIVITY_DETECTED, CF_CFDP_ConditionCode_NO_ERROR, CF_TxnStatus_ACK_LIMIT_NO_EOF, CF_TxnStatus_ACK_LIMIT_NO_FIN, CF_TxnStatus_CANCEL_REQUEST_RECEIVED, CF_TxnStatus_CHECK_LIMIT_REACHED, CF_TxnStatus_FILE_CHECKSUM_FAILURE, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_FILESTORE_REJECTION, CF_TxnStatus_INACTIVITY_DETECTED, CF_TxnStatus_INVALID_FILE_STRUCTURE, CF_TxnStatus_INVALID_TRANSMISSION_MODE, CF_TxnStatus_IsError(), CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED, CF_TxnStatus_NAK_LIMIT_REACHED, CF_TxnStatus_NO_ERROR, CF_TxnStatus_POS_ACK_LIMIT_REACHED, CF_TxnStatus_SUSPEND_REQUEST_RECEIVED, and CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE.

Referenced by CF_CFDP_R2_SubstateSendFin(), and CF_CFDP_SendEof().

Here is the call graph for this function:



12.49.2.16 CF_WrappedClose() void CF_WrappedClose (

```
    osal_id_t fd )
```

Wrap the filesystem close call with a perf counter.

Assumptions, External Events, and Notes:

None

See also

[OS_close\(\)](#) for parameter descriptions

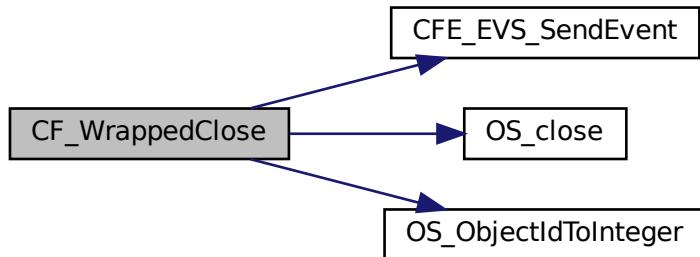
Parameters

<i>fd</i>	Passed directly to underlying OSAL call
-----------	---

Definition at line 437 of file cf_utils.c.

References CF_CFDP_CLOSE_ERR_EID, CF_PERF_ID_FCLOSE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), OS_close(), OS_ObjectIdToInteger(), and OS_SUCCESS. Referenced by CF_CFDP_CloseFiles(), CF_CFDP_R2_RecvMd(), CF_CFDP_ResetTransaction(), and CF_WriteQueueCmd().

Here is the call graph for this function:



12.49.2.17 CF_WrappedLseek() [CFE_Status_t](#) CF_WrappedLseek (
 osal_id_t *fd*,
 off_t *offset*,
 int *mode*)

Wrap the filesystem lseek call with a perf counter.

Assumptions, External Events, and Notes:

fname must not be NULL.

See also

[OS_lseek\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
-----------	---

Parameters

<i>offset</i>	Passed directly to underlying OSAL call
<i>mode</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 490 of file cf_utils.c.

References CF_PERF_ID_FSEEK, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_Iseek().

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R_ProcessFd(), CF_CFDP_S_SendFileData(), and CF_CFDP_S_SubstateSendMetadata().

Here is the call graph for this function:



12.49.2.18 CF_WrappedOpenCreate() `CFE_Status_t CF_WrappedOpenCreate (`
`osal_id_t * fd,`
`const char * fname,`
`int32 flags,`
`int32 access)`

Wrap the filesystem open call with a perf counter.

Assumptions, External Events, and Notes:

fname must not be NULL.

See also

[OS_OpenCreate\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>fname</i>	Passed directly to underlying OSAL call
<i>flags</i>	Passed directly to underlying OSAL call
<i>access</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL

Definition at line 421 of file cf_utils.c.

References CF_PERF_ID_FOPEN, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_OpenCreate().

Referenced by CF_CFDP_R2_RecvMd(), CF_CFDP_R_Init(), CF_CFDP_S_SubstateSendMetadata(), and CF_WriteQueueCmd().

Here is the call graph for this function:

**12.49.2.19 CF_WrappedRead()** [CFE_Status_t](#) CF_WrappedRead (

```
    osal_id_t fd,  
    void * buf,  
    size_t read_size )
```

Wrap the filesystem read call with a perf counter.

Assumptions, External Events, and Notes:

buf must not be NULL.

See also

[OS_read\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>read_size</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 458 of file cf_utils.c.

References CF_PERF_ID_FREAD, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_read().

Referenced by CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_S_SendFileData().

Here is the call graph for this function:



12.49.2.20 CF_WrappedWrite() [CFE_Status_t](#) CF_WrappedWrite (

```
    osal_id_t fd,  
    const void * buf,  
    size_t write_size )
```

Wrap the filesystem write call with a perf counter.

Assumptions, External Events, and Notes:

buf must not be NULL.

See also

[OS_write\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>write_size</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 474 of file cf_utils.c.

References CF_PERF_ID_FWRITE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_write().

Referenced by CF_CFDP_R_ProcessFd(), and CF_WriteHistoryEntryToFile().

Here is the call graph for this function:



12.49.2.21 CF_WriteHistoryEntryToFile() `CFE_Status_t CF_WriteHistoryEntryToFile (`
 `osal_id_t fd,`
 `const CF_History_t * history)`

Write a single history to a file.

This creates a human-readable/string representation of the information in the history object, and writes that string to the indicated file.

This implements the common code between writing the history queue and transaction queue to a file, as both ultimately store the same information in a CF_History_t object, but have a different method to get to it.

Assumptions, External Events, and Notes:

fd should be a valid file descriptor, open for writing.

Parameters

<i>fd</i>	Open File descriptor to write to
<i>history</i>	Pointer to CF history object to write

Return values

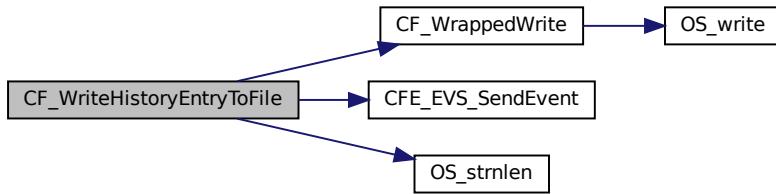
<i>CFE_SUCCESS</i>	on success
<i>CF_ERROR</i>	on error

Definition at line 172 of file cf_utils.c.

References CF_ASSERT, CF_CMD_WHIST_WRITE_ERR_EID, CF_Direction_NUM, CF_ERROR, CF_FILENAME_MAX_LEN, CF_WrappedWrite(), CFE_EVT_EventType_ERROR, CFE_EVT_SendEvent(), CFE_SUCCESS, CF_History::dir, CF_TxnFilenames::dst_filename, CF_History::fnames, OS_strlen(), CF_History::peer_eid, CF_History::seq_num, CF_History::src_eid, CF_TxnFilenames::src_filename, and CF_History::txn_stat.

Referenced by CF_Traverse_WriteHistoryQueueToFile(), and CF_Traverse_WriteTxnQueueEntryToFile().

Here is the call graph for this function:



12.49.2.22 CF_WriteHistoryQueueDataToFile() CFE_Status_t CF_WriteHistoryQueueDataToFile (

```

osal_id_t fd,
CF_Channel_t * chan,
CF_Direction_t dir )

```

Write a history-based queue's entries to a file.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>dir</i>	Direction to match/filter

Return values

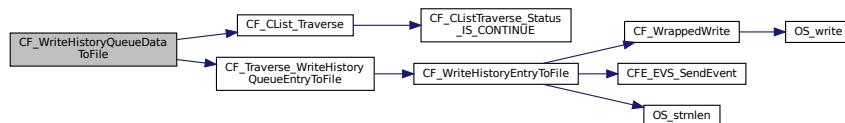
0	on success
1	on error

Definition at line 289 of file cf_utils.c.

References CF_CList_Traverse(), CF_QueueIdx_HIST, CF_Traverse_WriteHistoryQueueEntryToFile(), CF_Traverse<->_WriteHistoryFileArg::counter, CF_Traverse_WriteHistoryFileArg::error, CF_Traverse_WriteHistoryFileArg::fd, CF_Traverse<->_WriteHistoryFileArg::filter_dir, and CF_Channel::qs.

Referenced by CF_WriteQueueCmd().

Here is the call graph for this function:



```
12.49.2.23 CF_WriteTxnQueueDataToFile() CFE_Status_t CF_WriteTxnQueueDataToFile (
    osal_id_t fd,
    CF_Channel_t * chan,
    CF_QueueIdx_t queue )
```

Write a transaction-based queue's transaction history to a file.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>queue</i>	Queue Index to write

Return values

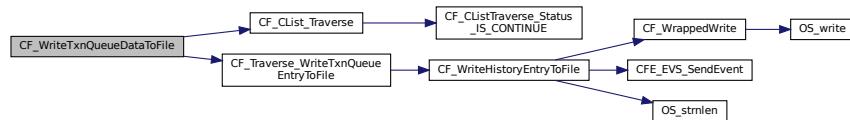
0	on success
1	on error

Definition at line 271 of file cf_utils.c.

References CF_CList_Traverse(), CF_Traverse_WriteTxnQueueEntryToFile(), CF_Traverse_WriteTxnFileArg::counter, CF_Traverse_WriteTxnFileArg::error, CF_Traverse_WriteTxnFileArg::fd, and CF_Channel::qs.

Referenced by CF_WriteQueueCmd().

Here is the call graph for this function:



12.50 apps/cf/fsw/src(cf_utils.h File Reference

```
#include "cf_cfdp.h"
#include "cf_app.h"
#include "cf_assert.h"
```

Data Structures

- struct `CF_Traverse_TransSeqArg`
Argument structure for use with CList_Traverse()
- struct `CF_Traverse_WriteHistoryFileArg`
Argument structure for use with CF_Traverse_WriteHistoryQueueEntryToFile()
- struct `CF_Traverse_WriteTxnFileArg`

- struct **CF_TraverseAll_Arg**
 - Argument structure for use with CF_Traverse_WriteTxnQueueEntryToFile()*
- struct **CF_Traverse_PriorityArg**
 - Argument structure for use with CF_TraverseAllTransactions()*
- struct **CF_CList_Traverse_R()**
 - Argument structure for use with CF_CList_Traverse_R()*

Typedefs

- typedef struct **CF_Traverse_TransSeqArg** **CF_Traverse_TransSeqArg_t**
 - Argument structure for use with CList_Traverse()*
- typedef struct **CF_Traverse_WriteHistoryFileArg** **CF_Traverse_WriteHistoryFileArg_t**
 - Argument structure for use with CF_Traverse_WriteHistoryQueueEntryToFile()*
- typedef struct **CF_Traverse_WriteTxnFileArg** **CF_Traverse_WriteTxnFileArg_t**
 - Argument structure for use with CF_Traverse_WriteTxnQueueEntryToFile()*
- typedef void(* **CF_TraverseAllTransactions_fn_t**) (**CF_Transaction_t** *txn, void *context)
 - Callback function type for use with CF_TraverseAllTransactions()*
- typedef struct **CF_TraverseAll_Arg** **CF_TraverseAll_Arg_t**
 - Argument structure for use with CF_TraverseAllTransactions()*
- typedef struct **CF_Traverse_PriorityArg** **CF_Traverse_PriorityArg_t**
 - Argument structure for use with CF_CList_Traverse_R()*

Functions

- static void **CF_DequeueTransaction** (**CF_Transaction_t** *txn)
- static void **CF_MoveTransaction** (**CF_Transaction_t** *txn, **CF_QueueIdx_t** queue)
- static void **CF_CList_Remove_Ex** (**CF_Channel_t** *chan, **CF_QueueIdx_t** queueidx, **CF_CListNode_t** *node)
- static void **CF_CList_InsertAfter_Ex** (**CF_Channel_t** *chan, **CF_QueueIdx_t** queueidx, **CF_CListNode_t** *start, **CF_CListNode_t** *after)
- static void **CF_CList_InsertBack_Ex** (**CF_Channel_t** *chan, **CF_QueueIdx_t** queueidx, **CF_CListNode_t** *node)
- **CF_Transaction_t** * **CF_FindUnusedTransaction** (**CF_Channel_t** *chan)
 - Find an unused transaction on a channel.*
- void **CF_ResetHistory** (**CF_Channel_t** *chan, **CF_History_t** *history)
 - Returns a history structure back to its unused state.*
- void **CF_FreeTransaction** (**CF_Transaction_t** *txn)
 - Frees and resets a transaction and returns it for later use.*
- **CF_Transaction_t** * **CF_FindTransactionBySequenceNumber** (**CF_Channel_t** *chan, **CF_TransactionSeq_t** transaction_sequence_number, **CF_EntityId_t** src_eid)
 - Finds an active transaction by sequence number.*
- **CFE_Status_t** **CF_FindTransactionBySequenceNumber_Impl** (**CF_CListNode_t** *node, **CF_Traverse_TransSeqArg_t** *context)
 - List traversal function to check if the desired sequence number matches.*
- **CFE_Status_t** **CF_WriteHistoryEntryToFile** (**osal_id_t** fd, const **CF_History_t** *history)
 - Write a single history to a file.*
- **CFE_Status_t** **CF_WriteTxnQueueDataToFile** (**osal_id_t** fd, **CF_Channel_t** *chan, **CF_QueueIdx_t** queue)
 - Write a transaction-based queue's transaction history to a file.*
- **CFE_Status_t** **CF_WriteHistoryQueueDataToFile** (**osal_id_t** fd, **CF_Channel_t** *chan, **CF_Direction_t** dir)
 - Write a history-based queue's entries to a file.*
- void **CF_InsertSortPrio** (**CF_Transaction_t** *txn, **CF_QueueIdx_t** queue)
 - Insert a transaction into a priority sorted transaction queue.*

- `int32 CF_TraverseAllTransactions (CF_Channel_t *chan, CF_TraverseAllTransactions_fn_t fn, void *context)`
Traverses all transactions on all active queues and performs an operation on them.
- `int32 CF_TraverseAllTransactions_All_Channels (CF_TraverseAllTransactions_fn_t fn, void *context)`
Traverses all transactions on all channels and performs an operation on them.
- `CF_CListTraverse_Status_t CF_TraverseAllTransactions_Impl (CF_CListNode_t *node, void *arg)`
List traversal function performs operation on every active transaction.
- `CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile (CF_CListNode_t *node, void *arg)`
Writes a human readable representation of a history queue entry to a file.
- `CF_CListTraverse_Status_t CF_Traverse_WriteTxnQueueEntryToFile (CF_CListNode_t *node, void *arg)`
Writes a human readable representation of a transaction history entry to a file.
- `CF_CListTraverse_Status_t CF_PrioSearch (CF_CListNode_t *node, void *context)`
Searches for the first transaction with a lower priority than given.
- `CFE_Status_t CF_WrappedOpenCreate (osal_id_t *fd, const char *fname, int32 flags, int32 access)`
Wrap the filesystem open call with a perf counter.
- `void CF_WrappedClose (osal_id_t fd)`
Wrap the filesystem close call with a perf counter.
- `CFE_Status_t CF_WrappedRead (osal_id_t fd, void *buf, size_t read_size)`
Wrap the filesystem read call with a perf counter.
- `CFE_Status_t CF_WrappedWrite (osal_id_t fd, const void *buf, size_t write_size)`
Wrap the filesystem write call with a perf counter.
- `CFE_Status_t CF_WrappedLseek (osal_id_t fd, off_t offset, int mode)`
Wrap the filesystem lseek call with a perf counter.
- `CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode (CF_TxnStatus_t txn_stat)`
Converts the internal transaction status to a CFDP condition code.
- `CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (CF_CFDP_ConditionCode_t cc)`
Converts a CFDP condition code to an internal transaction status.
- `bool CF_TxnStatus_IsError (CF_TxnStatus_t txn_stat)`
Check if the internal transaction status represents an error.

12.50.1 Detailed Description

The CF Application utils header file

12.50.2 Typedef Documentation

12.50.2.1 CF_Traverse_PriorityArg_t `typedef struct CF_Traverse_PriorityArg CF_Traverse_PriorityArg_t`
Argument structure for use with `CF_CList_Traverse_R()`
This is for searching for transactions of a specific priority

12.50.2.2 CF_Traverse_TransSeqArg_t `typedef struct CF_Traverse_TransSeqArg CF_Traverse_TransSeqArg_t`
Argument structure for use with `CList_Traverse()`
This identifies a specific transaction sequence number and entity ID The transaction pointer is set by the implementation

12.50.2.3 CF_Traverse_WriteHistoryFileArg_t `typedef struct CF_Traverse_WriteHistoryFileArg CF_Traverse_WriteHistoryFileArg_t`
Argument structure for use with `CF_Traverse_WriteHistoryQueueEntryToFile()`
This is used for writing status files. It contains a designated file descriptor for output and counters.
When traversing history, the list contains all entries, and may need additional filtering for direction (TX/RX) depending on what information the user has requested.

12.50.2.4 CF_Traverse_WriteTxnFileArg_t `typedef struct CF_Traverse_WriteTxnFileArg CF_Traverse_WriteTxnFileArg_t`
 Argument structure for use with [CF_Traverse_WriteTxnQueueEntryToFile\(\)](#)

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing transactions, the entire list is written to the file. No additional filtering is necessary, because the queues themselves are limited in what they contain (therefore "pre-filtered" to some degree).

12.50.2.5 CF_TraverseAll_Arg_t `typedef struct CF_TraverseAll_Arg CF_TraverseAll_Arg_t`
 Argument structure for use with [CF_TraverseAllTransactions\(\)](#)

This basically allows for running a CF_Traverse on several lists at once

12.50.2.6 CF_TraverseAllTransactions_fn_t `typedef void(* CF_TraverseAllTransactions_fn_t) (CF_Transaction_t *tx, void *context)`

Callback function type for use with [CF_TraverseAllTransactions\(\)](#)

Parameters

<i>tx</i>	Pointer to current transaction being traversed
<i>context</i>	Opaque object passed from initial call

Definition at line 88 of file cf_utils.h.

12.50.3 Function Documentation

12.50.3.1 CF_CList_InsertAfter_Ex() `static void CF_CList_InsertAfter_Ex (`
`CF_Channel_t * chan,`
`CF_QueueIdx_t queueidx,`
`CF_CListNode_t * start,`
`CF_CListNode_t * after) [inline], [static]`

Definition at line 148 of file cf_utils.h.

References CF_AppData, CF_CList_InsertAfter(), CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::engine, CF_AppData_t::hk, CF_HkPacket::Payload, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_InsertSortPrio().

Here is the call graph for this function:

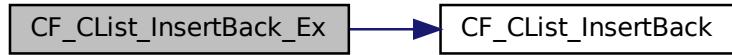


12.50.3.2 CF_CList_InsertBack_Ex() `static void CF_CList_InsertBack_Ex (`
`CF_Channel_t * chan,`
`CF_QueueIdx_t queueidx,`
`CF_CListNode_t * node) [inline], [static]`

Definition at line 155 of file cf_utils.h.

References CF_AppData, CF_CList_InsertBack(), CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::engine, CF_AppData_t::hk, CF_HkPacket::Payload, CF_HkChannel_Data::q_size, and CF_Channel::qs.
Referenced by CF_CFDP_InitEngine(), CF_CFDP_ReceiveMessage(), CF_CFDP_ResetTransaction(), CF_FreeTransaction(), CF_InsertSortPrio(), and CF_ResetHistory().

Here is the call graph for this function:



12.50.3.3 CF_CList_Remove_Ex() static void CF_CList_Remove_Ex (CF_Channel_t * chan, CF_QueueIdx_t queueidx, CF_CListNode_t * node) [inline], [static]

Definition at line 141 of file cf_utils.h.

References CF_AppData, CF_Assert, CF_CList_Remove(), CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_AppData_t::engine, CF_AppData_t::hk, CF_HkPacket::Payload, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_FindUnusedTransaction(), and CF_ResetHistory().

Here is the call graph for this function:



12.50.3.4 CF_DequeueTransaction() static void CF_DequeueTransaction (CF_Transaction_t * txn) [inline], [static]

Definition at line 122 of file cf_utils.h.

References CF_AppData, CF_Assert, CF_CList_Remove(), CF_NUM_CHANNELS, CF_Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::engine, CF_Transaction::flags, CF_AppData_t::hk, CF_HkPacket::Payload, CF_Flags_Common::q_index, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_CFDP_ResetTransaction(), and CF_CFDP_S2_SubstateSendEof().

Here is the call graph for this function:



12.50.3.5 CF_FindTransactionBySequenceNumber() `CF_Transaction_t* CF_FindTransactionBySequenceNumber (`

`CF_Channel_t * chan,`
`CF_TransactionSeq_t transaction_sequence_number,`
`CF_EntityId_t src_eid)`

Finds an active transaction by sequence number.

Description

This function traverses the active rx, pending, txa, and txw transaction and looks for the requested transaction.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<code>chan</code>	Pointer to the CF channel
<code>transaction_sequence_number</code>	Sequence number to find
<code>src_eid</code>	Entity ID associated with sequence number

Returns

Pointer to the given transaction if found

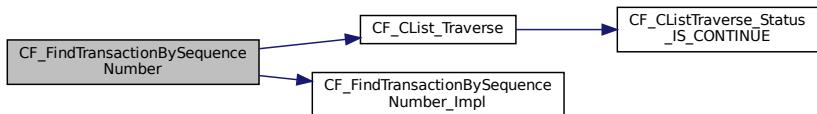
Return values

<code>NULL</code>	if the transaction is not found
-------------------	---------------------------------

Definition at line 139 of file cf_utils.c.

References `CF_CList_Traverse()`, `CF_FindTransactionBySequenceNumber_Impl()`, `CF_QueueIdx_PEND`, `CF_QueueIdx_RX`, `CF_QueueIdx_TXA`, `CF_QueueIdx_TXW`, `CF_Channel::qs`, and `CF_Traverse_TransSeqArg::txn`. Referenced by `CF_CFDP_ReceiveMessage()`, and `CF_FindTransactionBySequenceNumberAllChannels()`.

Here is the call graph for this function:



12.50.3.6 CF_FindTransactionBySequenceNumber_Impl() `CFE_Status_t CF_FindTransactionBySequenceNumber_Impl(`

`CF_CListNode_t * node,
CF_Traverse_TransSeqArg_t * context)`

List traversal function to check if the desired sequence number matches.

Assumptions, External Events, and Notes:

context must not be NULL. node must not be NULL.

Parameters

<code>node</code>	Pointer to node currently being traversed
<code>context</code>	Pointer to state object passed through from initial call

Return values

<code>1</code>	when it's found, which terminates list traversal
<code>0</code>	when it isn't found, which causes list traversal to continue

Definition at line 119 of file cf_utils.c.

References container_of, CF_Transaction::history, CF_History::seq_num, CF_Traverse_TransSeqArg::src_eid, CF_History::src_eid, CF_Traverse_TransSeqArg::transaction_sequence_number, and CF_Traverse_TransSeqArg::txn. Referenced by CF_FindTransactionBySequenceNumber().

12.50.3.7 CF_FindUnusedTransaction() `CF_Transaction_t* CF_FindUnusedTransaction(`

`CF_Channel_t * chan)`

Find an unused transaction on a channel.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<code>chan</code>	Pointer to the CF channel
-------------------	---------------------------

Returns

Pointer to a free transaction

Return values

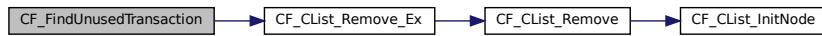
<code>NULL</code>	if no free transactions available.
-------------------	------------------------------------

Definition at line 43 of file cf_utils.c.

References `CF_ASSERT`, `CF_CList_Remove_Ex()`, `CF_Direction_NUM`, `CF_QueueIdx_FREE`, `CF_QueueIdx_HIST`, `CF_QueueIdx_HIST_FREE`, `CF_History::cl_node`, `CF_Transaction::cl_node`, `container_of`, `CF_History::dir`, `CF_Transaction::history`, and `CF_Channel::qs`.

Referenced by `CF_CFDP_ProcessPlaybackDirectory()`, `CF_CFDP_ReceiveMessage()`, and `CF_CFDP_TxFile()`.

Here is the call graph for this function:



12.50.3.8 CF_FreeTransaction()

```
void CF_FreeTransaction (
    CF_Transaction_t * txn )
```

Frees and resets a transaction and returns it for later use.

Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

Parameters

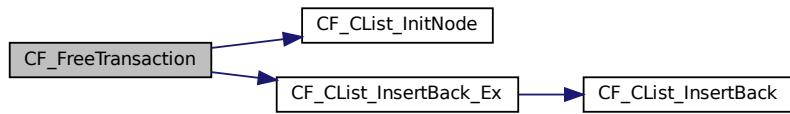
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 101 of file cf_utils.c.

References `CF_AppData`, `CF_CList_InitNode()`, `CF_CList_InsertBack_Ex()`, `CF_QueueIdx_FREE`, `CF_TxnState_ID`, `CF_Transaction::chan_num`, `CF_Engine::channels`, `CF_Transaction::cl_node`, `CF_StateFlags::com`, `CF_AppData::t::engine`, `CF_Transaction::fd`, `CF_Transaction::flags`, `OS_OBJECT_ID_UNDEFINED`, `CF_Flags_Common::q_index`, and `CF_Transaction::state`.

Referenced by `CF_CFDP_InitEngine()`, and `CF_CFDP_ResetTransaction()`.

Here is the call graph for this function:



```
12.50.3.9 CF_InsertSortPrio() void CF_InsertSortPrio (
    CF_Transaction_t * txn,
    CF_QueueIdx_t queue )
```

Insert a transaction into a priority sorted transaction queue.

Description

This function works by walking the queue in reverse to find a transaction with a higher priority than the given transaction. The given transaction is then inserted after that one, since it would be the next lower priority.

Assumptions, External Events, and Notes:

txn must not be NULL.

Parameters

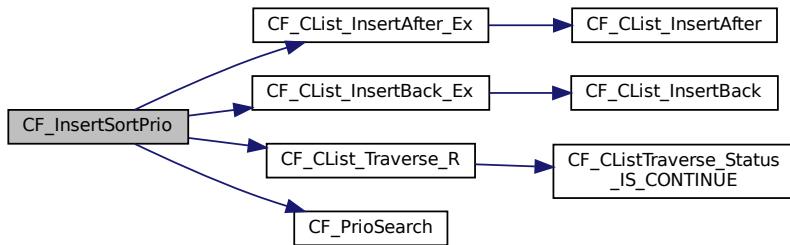
<i>txn</i>	Pointer to the transaction object
<i>queue</i>	Index of queue to insert into

Definition at line 332 of file cf_utils.c.

References CF_AppData, CF_Assert, CF_CList_InsertAfter_Ex(), CF_CList_InsertBack_Ex(), CF_CList_Traverse_R(), CF_NUM_CHANNELS, CF_PrioSearch(), CF_TxnState_IDLE, CF_Transaction::chan_num, CF_Engine::channels, C←F_Transaction::cl_node, CF_StateFlags::com, CF_AppData_t::engine, CF_Transaction::flags, CF_Transaction::priority, CF_Flags_Common::q_index, CF_Channel::qs, CF_Transaction::state, and CF_Traverse_PriorityArg::txn.

Referenced by CF_CFDP_S2_SubstateSendEof(), and CF_CFDP_TxFile_Initiate().

Here is the call graph for this function:



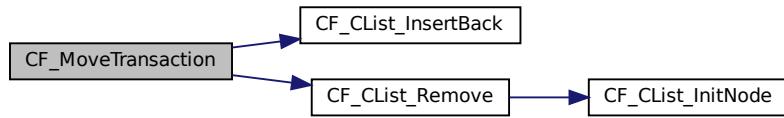
```
12.50.3.10 CF_MoveTransaction() static void CF_MoveTransaction (
    CF_Transaction_t * txn,
    CF_QueueIdx_t queue ) [inline], [static]
```

Definition at line 130 of file cf_utils.h.

References CF_AppData, CF_Assert, CF_CList_InsertBack(), CF_CList_Remove(), CF_NUM_CHANNELS, CF←Transaction::chan_num, CF_HkPacket_Payload::channel_hk, CF_Engine::channels, CF_Transaction::cl_node, CF←StateFlags::com, CF_AppData_t::engine, CF_Transaction::flags, CF_AppData_t::hk, CF_HkPacket::Payload, CF←Flags_Common::q_index, CF_HkChannel_Data::q_size, and CF_Channel::qs.

Referenced by CF_CFDP_CycleTx().

Here is the call graph for this function:



12.50.3.11 CF_PrioSearch() `CF_CListTraverse_Status_t CF_PrioSearch (`
`CF_CListNode_t * node,`
`void * context)`

Searches for the first transaction with a lower priority than given.

Assumptions, External Events, and Notes:

node must not be NULL. context must not be NULL.

Parameters

<code>node</code>	Node being currently traversed
<code>context</code>	Pointer to <code>CF_Traverse_PriorityArg_t</code> object indicating the priority to search for

Return values

<code>CF_CLIST_EXIT</code>	when it's found, which terminates list traversal
<code>CF_CLIST_CONT</code>	when it isn't found, which causes list traversal to continue

Definition at line 308 of file cf_utils.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `container_of`, `CF_Traverse_PriorityArg::priority`, `CF_Transaction::priority`, and `CF_Traverse_PriorityArg::txn`.

Referenced by `CF_InsertSortPrio()`.

12.50.3.12 CF_ResetHistory() `void CF_ResetHistory (`
`CF_Channel_t * chan,`
`CF_History_t * history)`

Returns a history structure back to its unused state.

Description

There's nothing to do currently other than remove the history from its current queue and put it back on `CF_QueueIdx_HIST_FREE`.

Assumptions, External Events, and Notes:

chan must not be NULL. history must not be NULL.

Parameters

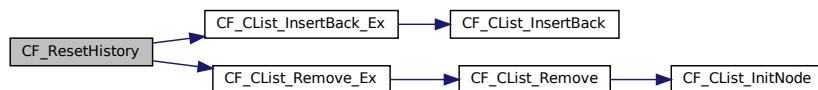
<i>chan</i>	Pointer to the CF channel
<i>history</i>	Pointer to the history entry

Definition at line 89 of file cf_utils.c.

References CF_CList_InsertBack_Ex(), CF_CList_Remove_Ex(), CF_QueueIdx_HIST, CF_QueueIdx_HIST_FREE, and CF_History::cl_node.

Referenced by CF_PurgeHistory().

Here is the call graph for this function:



12.50.3.13 CF_Traverse_WriteHistoryQueueEntryToFile()

`CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile (`

```

    CF_CListNode_t * node,
    void * arg )
  
```

Writes a human readable representation of a history queue entry to a file.

This function is a wrapper around [CF_WriteHistoryEntryToFile\(\)](#) that can be used with CF_Traverse() to write history queue entries to the file.

This also implements a direction filter, so only matching entries are actually written to the file.

Assumptions, External Events, and Notes:

node must not be NULL. arg must not be NULL.

Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to CF_Traverse_WriteHistoryFileArg_t indicating the file information

Return values

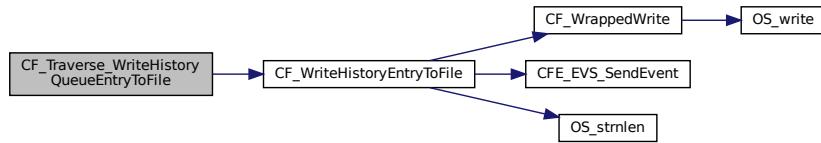
<i>CF_CLIST_CONT</i>	if everything is going well
<i>CF_CLIST_EXIT</i>	if a write error occurred, which means traversal should stop

Definition at line 222 of file cf_utils.c.

References CF_CLIST_CONT, CF_CLIST_EXIT, CF_Direction_NUM, CF_WriteHistoryEntryToFile(), container_of, CF_Traverse_WriteHistoryFileArg::counter, CF_History::dir, CF_Traverse_WriteHistoryFileArg::error, CF_Traverse_WriteHistoryFileArg::fd, and CF_Traverse_WriteHistoryFileArg::filter_dir.

Referenced by CF_WriteHistoryQueueDataToFile().

Here is the call graph for this function:



12.50.3.14 CF_Traverse_WriteTxnQueueEntryToFile() CF_CListTraverse_Status_t CF_Traverse_WriteTxn→

```
QueueEntryToFile (
    CF_CListNode_t * node,
    void * arg )
```

Writes a human readable representation of a transaction history entry to a file.

This function is a wrapper around [CF_WriteHistoryEntryToFile\(\)](#) that can be used with CF_Traverse() to write transaction queue entries to the file.

Assumptions, External Events, and Notes:

node must not be NULL. arg must not be NULL.

Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to CF_Traverse_WriteTxnFileArg_t indicating the file information

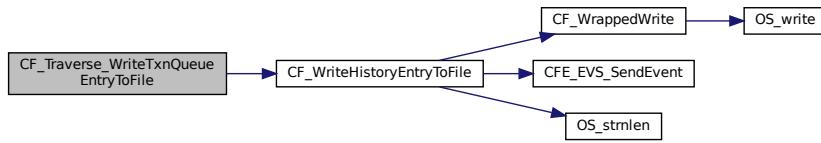
Return values

<i>CF_CLIST_CONT</i>	if everything is going well
<i>CF_CLIST_EXIT</i>	if a write error occurred, which means traversal should stop

Definition at line 249 of file cf_utils.c.

References CF_CLIST_CONT, CF_CLIST_EXIT, CF_WriteHistoryEntryToFile(), container_of, CF_Traverse_WriteTxnToFileArg::counter, CF_Traverse_WriteTxnFileArg::error, CF_Traverse_WriteTxnFileArg::fd, and CF_Transaction::history. Referenced by CF_WriteTxnQueueDataToFile().

Here is the call graph for this function:



```
12.50.3.15 CF_TraverseAllTransactions() int32 CF_TraverseAllTransactions (
    CF_Channel_t * chan,
    CF_TraverseAllTransactions_fn_t fn,
    void * context )
```

Traverses all transactions on all active queues and performs an operation on them.

Assumptions, External Events, and Notes:

chan must not be NULL. fn must be a valid function. context must not be NULL.

Parameters

<i>chan</i>	Channel to operate on
<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

Returns

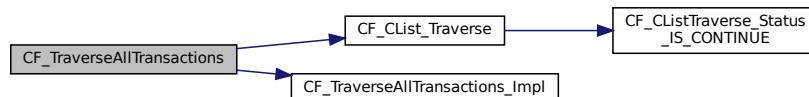
Number of transactions traversed

Definition at line 390 of file cf_utils.c.

References CF_CList_Traverse(), CF_QueueIdx_PEND, CF_QueueIdx_RX, CF_TraverseAllTransactions_Impl(), CF_TraverseAll_Arg::counter, and CF_Channel::qs.

Referenced by CF_TraverseAllTransactions_All_Channels(), and CF_TsnChanAction().

Here is the call graph for this function:



```
12.50.3.16 CF_TraverseAllTransactions_All_Channels() int32 CF_TraverseAllTransactions_All_Channels (
(
    CF_TraverseAllTransactions_fn_t fn,
    void * context )
```

Traverses all transactions on all channels and performs an operation on them.

Assumptions, External Events, and Notes:

fn must be a valid function. context must not be NULL.

Parameters

<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

Returns

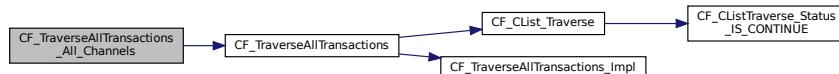
Number of transactions traversed

Definition at line 406 of file cf_utils.c.

References CF_AppData, CF_NUM_CHANNELS, CF_TraverseAllTransactions(), CF_Engine::channels, and CF_AppData_t::engine.

Referenced by CF_TsnChanAction().

Here is the call graph for this function:



12.50.3.17 CF_TraverseAllTransactions_Impl()

`CF_CListTraverse_Status_t CF_TraverseAllTransactions->_Impl (`

```
        CF_CListNode_t * node,
        void * arg )
```

List traversal function performs operation on every active transaction.

Description

Called on every transaction via list traversal. Calls another function on that transaction.

Assumptions, External Events, and Notes:

node must not be NULL. args must not be NULL.

Parameters

<code>node</code>	Node being currently traversed
<code>arg</code>	Intermediate context object from initial call

Return values

<code>0</code>	for do not exit early (always continue)
----------------	---

Definition at line 375 of file cf_utils.c.

References CF_CLIST_CONT, container_of, CF_TraverseAll_Arg::context, CF_TraverseAll_Arg::counter, and CF_TraverseAll_Arg::fn.

Referenced by CF_TraverseAllTransactions().

12.50.3.18 CF_TxnStatus_From_ConditionCode()

`CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (`

```
        CF_CFDP_ConditionCode_t cc )
```

Converts a CFDP condition code to an internal transaction status.

Assumptions, External Events, and Notes:

None

Parameters

cc	CFDP condition code
----	---------------------

Returns

Transaction status code

Definition at line 589 of file cf_utils.c.

Referenced by CF_CFDP_R2_SubstateRecvEof().

12.50.3.19 CF_TxnStatus_IsError() `bool CF_TxnStatus_IsError (`
`CF_TxnStatus_t txn_stat)`

Check if the internal transaction status represents an error.

Assumptions, External Events, and Notes:

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

Parameters

txn_stat	Transaction status
----------	--------------------

Returns

Boolean value indicating if the transaction is in an errored state

Return values

true	if an error has occurred during the transaction
false	if no error has occurred during the transaction yet

Definition at line 507 of file cf_utils.c.

References CF_TxnStatus_NO_ERROR.

Referenced by CF_CFDP_HandleNotKeepFile(), CF_CFDP_R2_Complete(), CF_CFDP_R2_Reset(), CF_CFDP_R2_SubstateSendFin(), CF_CFDP_R_DispatchRecv(), CF_CFDP_S2_WaitForEofAck(), CF_CFDP_SetTxnStatus(), and CF_TxnStatus_To_ConditionCode().

12.50.3.20 CF_TxnStatus_To_ConditionCode() `CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode (`
`CF_TxnStatus_t txn_stat)`

Converts the internal transaction status to a CFDP condition code.

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

Assumptions, External Events, and Notes:

Not all transaction status codes directly correlate to a CFDP CC

Parameters

<i>txn_stat</i>	Transaction status
-----------------	--------------------

Returns

CFDP protocol condition code

Definition at line 523 of file cf_utils.c.

References CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED, CF_CFDP_ConditionCode_INACTIVITY_DETECTED, CF_CFDP_ConditionCode_NO_ERROR, CF_TxnStatus_ACK_LIMIT_NO_EOF, CF_TxnStatus_ACK_LIMIT_NO_FIN, CF_TxnStatus_CANCEL_REQUEST_RECEIVED, CF_TxnStatus_CHECK_LIMIT_REACHED, CF_TxnStatus_FILE_CHECKSUM_FAILURE, CF_TxnStatus_FILE_SIZE_ERROR, CF_TxnStatus_FILESTORE_REJECTION, CF_TxnStatus_INACTIVITY_DETECTED, CF_TxnStatus_INVALID_FILE_STRUCTURE, CF_TxnStatus_INVALID_TRANSMISSION_MODE, CF_TxnStatus_IsError(), CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED, CF_TxnStatus_NAK_LIMIT_REACHED, CF_TxnStatus_NO_ERROR, CF_TxnStatus_POS_ACK_LIMIT_REACHED, CF_TxnStatus_SUSPEND_REQUEST_RECEIVED, and CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE.
Referenced by CF_CFDP_R2_SubstateSendFin(), and CF_CFDP_SendEof().

Here is the call graph for this function:



12.50.3.21 CF_WrappedClose()

```
void CF_WrappedClose (
    osal_id_t fd )
```

Wrap the filesystem close call with a perf counter.

Assumptions, External Events, and Notes:

None

See also

[OS_close\(\)](#) for parameter descriptions

Parameters

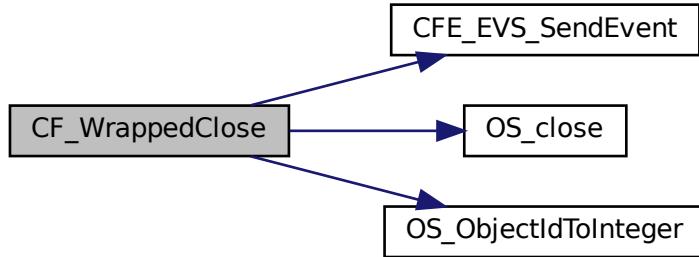
<i>fd</i>	Passed directly to underlying OSAL call
-----------	---

Definition at line 437 of file cf_utils.c.

References CF_CFDP_CLOSE_ERR_EID, CF_PERF_ID_FCLOSE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit,

CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), OS_close(), OS_ObjectIdToInteger(), and OS_SUCCESS. Referenced by CF_CFDP_CloseFiles(), CF_CFDP_R2_RecvMd(), CF_CFDP_ResetTransaction(), and CF_WriteQueueCmd().

Here is the call graph for this function:



12.50.3.22 CF_WrappedLseek()

```
CFE_Status_t CF_WrappedLseek (
    osal_id_t fd,
    off_t offset,
    int mode )
```

Wrap the filesystem lseek call with a perf counter.

Assumptions, External Events, and Notes:

fname must not be NULL.

See also

[OS_lseek\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>offset</i>	Passed directly to underlying OSAL call
<i>mode</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 490 of file cf_utils.c.

References CF_PERF_ID_FSEEK, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_Iseek().

Referenced by CF_CFDP_R2_CalcCrcChunk(), CF_CFDP_R_ProcessFd(), CF_CFDP_S_SendFileData(), and CF_CFDP_S_SubstateSendMetadata().

Here is the call graph for this function:

**12.50.3.23 CF_WrappedOpenCreate()** [CFE_Status_t](#) CF_WrappedOpenCreate (

```

    osal_id_t * fd,
    const char * fname,
    int32 flags,
    int32 access
  
```

Wrap the filesystem open call with a perf counter.

Assumptions, External Events, and Notes:

fname must not be NULL.

See also

[OS_OpenCreate\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>fname</i>	Passed directly to underlying OSAL call
<i>flags</i>	Passed directly to underlying OSAL call
<i>access</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL

Definition at line 421 of file cf_utils.c.

References CF_PERF_ID_FOPEN, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_OpenCreate().

Referenced by CF_CFDP_R2_RecvMd(), CF_CFDP_R_Init(), CF_CFDP_S_SubstateSendMetadata(), and CF_WriteQueueCmd().

Here is the call graph for this function:



12.50.3.24 CF_WrappedRead() [CFE_Status_t](#) CF_WrappedRead (

```
    osal_id_t fd,  
    void * buf,  
    size_t read_size )
```

Wrap the filesystem read call with a perf counter.

Assumptions, External Events, and Notes:

buf must not be NULL.

See also

[OS_read\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>read_size</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 458 of file cf_utils.c.

References CF_PERF_ID_FREAD, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_read().

Referenced by CF_CFDP_R2_CalcCrcChunk(), and CF_CFDP_S_SendFileData().

Here is the call graph for this function:

**12.50.3.25 CF_WrappedWrite()** [CFE_Status_t](#) CF_WrappedWrite (

```
    osal_id_t fd,  
    const void * buf,  
    size_t write_size )
```

Wrap the filesystem write call with a perf counter.

Assumptions, External Events, and Notes:

buf must not be NULL.

See also

[OS_write\(\)](#) for parameter descriptions

Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>write_size</i>	Passed directly to underlying OSAL call

Returns

Status code from OSAL (byte count or error code)

Definition at line 474 of file cf_utils.c.

References CF_PERF_ID_FWRITE, CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, and OS_write().

Referenced by CF_CFDP_R_ProcessFd(), and CF_WriteHistoryEntryToFile().

Here is the call graph for this function:



12.50.3.26 CF_WriteHistoryEntryToFile() `CFE_Status_t CF_WriteHistoryEntryToFile (`
 `osal_id_t fd,`
 `const CF_History_t * history)`

Write a single history to a file.

This creates a human-readable/string representation of the information in the history object, and writes that string to the indicated file.

This implements the common code between writing the history queue and transaction queue to a file, as both ultimately store the same information in a CF_History_t object, but have a different method to get to it.

Assumptions, External Events, and Notes:

fd should be a valid file descriptor, open for writing.

Parameters

<code>fd</code>	Open File descriptor to write to
<code>history</code>	Pointer to CF history object to write

Return values

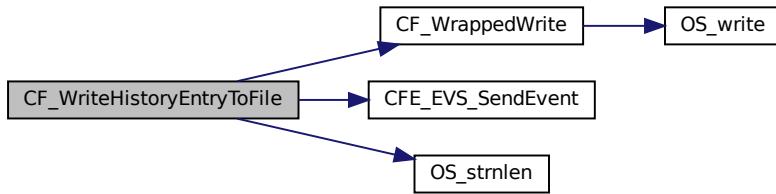
<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	on error

Definition at line 172 of file cf_utils.c.

References CF_ASSERT, CF_CMD_WHIST_WRITE_ERR_EID, CF_Direction_NUM, CF_ERROR, CF_FILENAME_MAX_LEN, CF_WrappedWrite(), CFE_EVT_EventType_ERROR, CFE_EVT_SendEvent(), CFE_SUCCESS, CF_History::dir, CF_TxnFilenames::dst_filename, CF_History::fnames, OS_strlen(), CF_History::peer_eid, CF_History::seq_num, CF_History::src_eid, CF_TxnFilenames::src_filename, and CF_History::txn_stat.

Referenced by CF_Traverse_WriteHistoryQueueToFile(), and CF_Traverse_WriteTxnQueueEntryToFile().

Here is the call graph for this function:



12.50.3.27 CF_WriteHistoryQueueDataToFile() CFE_Status_t CF_WriteHistoryQueueDataToFile (

```

osal_id_t fd,
CF_Channel_t * chan,
CF_Direction_t dir )

```

Write a history-based queue's entries to a file.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>dir</i>	Direction to match/filter

Return values

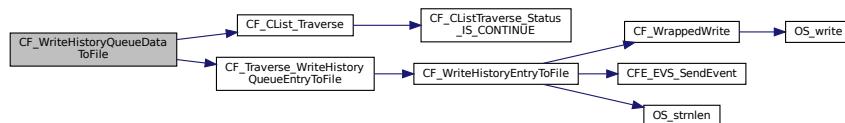
0	on success
1	on error

Definition at line 289 of file cf_utils.c.

References CF_CList_Traverse(), CF_QueueIdx_HIST, CF_Traverse_WriteHistoryQueueEntryToFile(), CF_Traverse<->_WriteHistoryFileArg::counter, CF_Traverse_WriteHistoryFileArg::error, CF_Traverse_WriteHistoryFileArg::fd, CF_Traverse<->_WriteHistoryFileArg::filter_dir, and CF_Channel::qs.

Referenced by CF_WriteQueueCmd().

Here is the call graph for this function:



```
12.50.3.28 CF_WriteTxnQueueDataToFile() CFE_Status_t CF_WriteTxnQueueDataToFile (
    osal_id_t fd,
    CF_Channel_t * chan,
    CF_QueueIdx_t queue )
```

Write a transaction-based queue's transaction history to a file.

Assumptions, External Events, and Notes:

chan must not be NULL.

Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>queue</i>	Queue Index to write

Return values

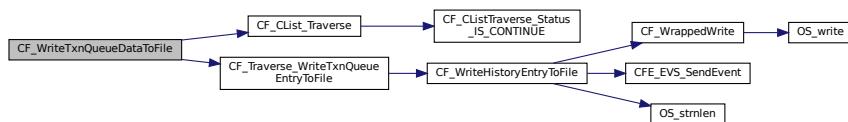
0	on success
1	on error

Definition at line 271 of file cf_utils.c.

References CF_CList_Traverse(), CF_Traverse_WriteTxnQueueEntryToFile(), CF_Traverse_WriteTxnFileArg::counter, CF_Traverse_WriteTxnFileArg::error, CF_Traverse_WriteTxnFileArg::fd, and CF_Channel::qs.

Referenced by CF_WriteQueueCmd().

Here is the call graph for this function:



12.51 apps/cf/fsw/src/cf_verify.h File Reference

```
#include "cfe.h"
#include "cf_platform_cfg.h"
#include "cf_perfids.h"
```

12.51.1 Detailed Description

The CF Application configuration verification header

12.52 apps/cf/fsw/src/cf_version.h File Reference

Macros

- #define CF_MAJOR_VERSION (3)

- `#define CF_MINOR_VERSION (0)`
Minor version number.
- `#define CF_REVISION (99)`
Revision number.

12.52.1 Detailed Description

The CFS CFDP (CF) Application header file containing version number

12.53 apps/cf/fsw/tables/cf_def_config.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "cf_tbldefs.h"
```

Variables

- `CF_ConfigTable_t CF_config_table`

12.53.1 Detailed Description

The CF Application default configuration table

12.53.2 Variable Documentation

12.53.2.1 CF_config_table `CF_ConfigTable_t CF_config_table`

Definition at line 28 of file cf_def_config.c.

12.54 buildosal_public_api/inc/osconfig.h File Reference

Macros

- `#define OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER`
Configuration file Operating System Abstraction Layer.
- `#define OSAL_CONFIG_INCLUDE_NETWORK`
- `#define OSAL_CONFIG_INCLUDE_STATIC_LOADER`
- `#define OSAL_CONFIG_CONSOLE_ASYNC`
- `#define OS_MAX_TASKS 64`
The maximum number of tasks to support.
- `#define OS_MAX_QUEUES 64`
The maximum number of queues to support.
- `#define OS_MAX_COUNT_SEMAPHORES 20`
The maximum number of counting semaphores to support.
- `#define OS_MAX_BIN_SEMAPHORES 20`
The maximum number of binary semaphores to support.
- `#define OS_MAX_MUTEXES 20`
The maximum number of mutexes to support.
- `#define OS_MAX_CONDVARS 4`

- `#define OS_MAX_MODULES 20`
The maximum number of modules to support.
- `#define OS_MAX_TIMEBASES 5`
The maximum number of timebases to support.
- `#define OS_MAX_TIMERS 10`
The maximum number of timer callbacks to support.
- `#define OS_MAX_NUM_OPEN_FILES 50`
The maximum number of concurrently open files to support.
- `#define OS_MAX_NUM_OPEN_DIRS 4`
The maximum number of concurrently open directories to support.
- `#define OS_MAX_FILE_SYSTEMS 14`
The maximum number of file systems to support.
- `#define OS_MAX_SYM_LEN 64`
The maximum length of symbols.
- `#define OS_MAX_FILE_NAME 20`
The maximum length of OSAL file names.
- `#define OS_MAX_PATH_LEN 64`
The maximum length of OSAL path names.
- `#define OS_MAX_API_NAME 20`
The maximum length of OSAL resource names.
- `#define OS_SOCKADDR_MAX_LEN 28`
The maximum size of the socket address structure.
- `#define OS_BUFFER_SIZE 172`
The maximum size of output produced by a single `OS_printf()`
- `#define OS_BUFFER_MSG_DEPTH 100`
The maximum number of `OS_printf()` output strings to buffer.
- `#define OS_UTILITYTASK_PRIORITY 245`
Priority level of the background utility task.
- `#define OS_UTILITYTASK_STACK_SIZE 2048`
The stack size of the background utility task.
- `#define OS_MAX_CMD_LEN 1000`
The maximum size of a shell command.
- `#define OS_QUEUE_MAX_DEPTH 50`
The maximum depth of OSAL queues.
- `#define OS_SHELL_CMD_INPUT_FILE_NAME ""`
The name of the temporary file used to store shell commands.
- `#define OS_PRINTF_CONSOLE_NAME ""`
The name of the primary console device.
- `#define OS_ADD_TASK_FLAGS 0`
Flags added to all tasks on creation.
- `#define OS_MAX_CONSOLES 1`
The maximum number of console devices to support.
- `#define OS_MODULE_FILE_EXTENSION ".so"`
The system-specific file extension used on loadable module files.
- `#define OS_FS_DEV_NAME_LEN 32`
- `#define OS_FS_PHYS_NAME_LEN 64`
- `#define OS_FS_VOL_NAME_LEN 32`

12.54.1 Macro Definition Documentation

12.54.1.1 OS_ADD_TASK_FLAGS #define OS_ADD_TASK_FLAGS 0

Flags added to all tasks on creation.

Added to the task flags on creation

Supports adding floating point support for all tasks when the OS requires it

Definition at line 254 of file osconfig.h.

12.54.1.2 OS_BUFFER_MSG_DEPTH #define OS_BUFFER_MSG_DEPTH 100

The maximum number of [OS_printf\(\)](#) output strings to buffer.

Based on the OSAL_CONFIG_PRINTF_BUFFER_DEPTH configuration option

Definition at line 187 of file osconfig.h.

12.54.1.3 OS_BUFFER_SIZE #define OS_BUFFER_SIZE 172

The maximum size of output produced by a single [OS_printf\(\)](#)

Based on the OSAL_CONFIG_PRINTF_BUFFER_SIZE configuration option

Definition at line 180 of file osconfig.h.

12.54.1.4 OS_FS_DEV_NAME_LEN #define OS_FS_DEV_NAME_LEN 32

Device name length

Definition at line 281 of file osconfig.h.

12.54.1.5 OS_FS_PHYS_NAME_LEN #define OS_FS_PHYS_NAME_LEN 64

Physical drive name length

Definition at line 282 of file osconfig.h.

12.54.1.6 OS_FS_VOL_NAME_LEN #define OS_FS_VOL_NAME_LEN 32

Volume name length

Definition at line 283 of file osconfig.h.

12.54.1.7 OS_MAX_API_NAME #define OS_MAX_API_NAME 20

The maximum length of OSAL resource names.

Based on the OSAL_CONFIG_MAX_API_NAME configuration option

Note

This value must include a terminating NUL character

Definition at line 163 of file osconfig.h.

12.54.1.8 OS_MAX_BIN_SEMAPHORES #define OS_MAX_BIN_SEMAPHORES 20

The maximum number of binary semaphores to support.

Based on the OSAL_CONFIG_MAX_BIN_SEMAPHORES configuration option

Definition at line 65 of file osconfig.h.

12.54.1.9 OS_MAX_CMD_LEN #define OS_MAX_CMD_LEN 1000

The maximum size of a shell command.

This limit is only applicable if shell support is enabled.

Based on the OSAL_CONFIG_MAX_CMD_LEN configuration option

Note

This value must include a terminating NUL character

Definition at line 218 of file osconfig.h.

12.54.1.10 OS_MAX_CONDVARS #define OS_MAX_CONDVARS 4

The maximum number of condition variables to support.

Based on the OSAL_CONFIG_MAX_CONDVARS configuration option

Definition at line 79 of file osconfig.h.

12.54.1.11 OS_MAX_CONSOLES #define OS_MAX_CONSOLES 1

The maximum number of console devices to support.

Fixed value based on current OSAL implementation, not user configurable.

Definition at line 269 of file osconfig.h.

12.54.1.12 OS_MAX_COUNT_SEMAPHORES #define OS_MAX_COUNT_SEMAPHORES 20

The maximum number of counting semaphores to support.

Based on the OSAL_CONFIG_MAX_COUNT_SEMAPHORES configuration option

Definition at line 58 of file osconfig.h.

12.54.1.13 OS_MAX_FILE_NAME #define OS_MAX_FILE_NAME 20

The maximum length of OSAL file names.

This limit applies specifically to the file name portion, not the directory portion, of a path name.

Based on the OSAL_CONFIG_MAX_FILE_NAME configuration option

Note

This value must include a terminating NUL character

Definition at line 142 of file osconfig.h.

12.54.1.14 OS_MAX_FILE_SYSTEMS #define OS_MAX_FILE_SYSTEMS 14

The maximum number of file systems to support.

Based on the OSAL_CONFIG_MAX_FILE_SYSTEMS configuration option

Definition at line 121 of file osconfig.h.

12.54.1.15 OS_MAX_MODULES #define OS_MAX_MODULES 20

The maximum number of modules to support.

Based on the OSAL_CONFIG_MAX_MODULES configuration option

Definition at line 86 of file osconfig.h.

12.54.1.16 OS_MAX_MUTEXES #define OS_MAX_MUTEXES 20

The maximum number of mutexes to support.

Based on the OSAL_CONFIG_MAX_MUTEXES configuration option

Definition at line 72 of file osconfig.h.

12.54.1.17 OS_MAX_NUM_OPEN_DIRS #define OS_MAX_NUM_OPEN_DIRS 4

The maximum number of concurrently open directories to support.

Based on the OSAL_CONFIG_MAX_NUM_OPEN_DIRS configuration option

Definition at line 114 of file osconfig.h.

12.54.1.18 OS_MAX_NUM_OPEN_FILES #define OS_MAX_NUM_OPEN_FILES 50

The maximum number of concurrently open files to support.

Based on the OSAL_CONFIG_MAX_NUM_OPEN_FILES configuration option

Definition at line 107 of file osconfig.h.

12.54.1.19 OS_MAX_PATH_LEN #define OS_MAX_PATH_LEN 64

The maximum length of OSAL path names.

This limit applies to the overall length of a path name, including the file name and directory portions.

Based on the OSAL_CONFIG_MAX_PATH_LEN configuration option

Note

This value must include a terminating NUL character

Definition at line 154 of file osconfig.h.

12.54.1.20 OS_MAX_QUEUES #define OS_MAX_QUEUES 64

The maximum number of queues to support.

Based on the OSAL_CONFIG_MAX_QUEUES configuration option

Definition at line 51 of file osconfig.h.

12.54.1.21 OS_MAX_SYM_LEN #define OS_MAX_SYM_LEN 64

The maximum length of symbols.

Based on the OSAL_CONFIG_MAX_SYM_LEN configuration option

Note

This value must include a terminating NUL character

Definition at line 130 of file osconfig.h.

12.54.1.22 OS_MAX_TASKS #define OS_MAX_TASKS 64

The maximum number of tasks to support.

Based on the OSAL_CONFIG_MAX_TASKS configuration option

Definition at line 44 of file osconfig.h.

12.54.1.23 OS_MAX_TIMEBASES #define OS_MAX_TIMEBASES 5

The maximum number of timebases to support.

Based on the OSAL_CONFIG_MAX_TIMEBASES configuration option

Definition at line 93 of file osconfig.h.

12.54.1.24 OS_MAX_TIMERS #define OS_MAX_TIMERS 10

The maximum number of timer callbacks to support.

Based on the OSAL_CONFIG_MAX_TIMERS configuration option

Definition at line 100 of file osconfig.h.

12.54.1.25 OS_MODULE_FILE_EXTENSION #define OS_MODULE_FILE_EXTENSION ".so"

The system-specific file extension used on loadable module files.

Fixed value based on system selection, not user configurable.

Definition at line 276 of file osconfig.h.

12.54.1.26 OS_PRINTF_CONSOLE_NAME #define OS_PRINTF_CONSOLE_NAME ""

The name of the primary console device.

This is the device to which [OS_Printf\(\)](#) output is written. The output may be configured to tag each line with this prefix for identification.

Based on the OSAL_CONFIG_PRINTF_CONSOLE_NAME configuration option

Definition at line 245 of file osconfig.h.

12.54.1.27 OS_QUEUE_MAX_DEPTH #define OS_QUEUE_MAX_DEPTH 50

The maximum depth of OSAL queues.

Based on the OSAL_CONFIG_QUEUE_MAX_DEPTH configuration option

Definition at line 225 of file osconfig.h.

12.54.1.28 OS_SHELL_CMD_INPUT_FILE_NAME #define OS_SHELL_CMD_INPUT_FILE_NAME ""

The name of the temporary file used to store shell commands.

This configuration is only applicable if shell support is enabled, and only necessary/relevant on some OS implementations.

Based on the OSAL_CONFIG_SHELL_CMD_INPUT_FILE_NAME configuration option

Definition at line 235 of file osconfig.h.

12.54.1.29 OS SOCKADDR_MAX_LEN #define OS SOCKADDR_MAX_LEN 28

The maximum size of the socket address structure.

This is part of the Socket API, and should be set large enough to hold the largest address type in use on the target system.

Based on the OSAL_CONFIG_SOCKADDR_MAX_LEN configuration option

Definition at line 173 of file osconfig.h.

12.54.1.30 OS UTILITYTASK_PRIORITY #define OS UTILITYTASK_PRIORITY 245

Priority level of the background utility task.

This task is responsible for writing buffered output of OS_Printf to the actual console device, and any other future maintenance task.

Based on the OSAL_CONFIG_UTILITYTASK_PRIORITY configuration option
Definition at line 197 of file osconfig.h.

12.54.1.31 OS_UTILITYTASK_STACK_SIZE #define OS_UTILITYTASK_STACK_SIZE 2048

The stack size of the background utility task.

This task is responsible for writing buffered output of OS_printf to the actual console device, and any other future maintenance task.

Based on the OSAL_CONFIG_UTILITYTASK_STACK_SIZE configuration option

Definition at line 207 of file osconfig.h.

12.54.1.32 OSAL_CONFIG_CONSOLE_ASYNC #define OSAL_CONFIG_CONSOLE_ASYNC

Definition at line 27 of file osconfig.h.

12.54.1.33 OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER #define OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER

Configuration file Operating System Abstraction Layer.

The specific definitions in this file may only be modified by setting the respective OSAL configuration options in the CMake build.

Any direct modifications to the generated copy will be overwritten each time CMake executes.

Note

This file was automatically generated by CMake from /home/runner/work/CF/CF/osal/default_config.cmake

Definition at line 21 of file osconfig.h.

12.54.1.34 OSAL_CONFIG_INCLUDE_NETWORK #define OSAL_CONFIG_INCLUDE_NETWORK

Definition at line 22 of file osconfig.h.

12.54.1.35 OSAL_CONFIG_INCLUDE_STATIC_LOADER #define OSAL_CONFIG_INCLUDE_STATIC_LOADER

Definition at line 23 of file osconfig.h.

12.55 cfe/cmake/sample_defs/example_mission_cfg.h File Reference

Macros

- #define CFE_MISSION_MAX_PATH_LEN 64
- #define CFE_MISSION_MAX_FILE_LEN 20
- #define CFE_MISSION_MAX_API_LEN 20
- #define CFE_MISSION_MAX_NUM_FILES 50
- #define CFE_MISSION_ES_MAX_APPLICATIONS 16
- #define CFE_MISSION_ES_PERF_MAX_IDS 128
- #define CFE_MISSION_ES_POOL_MAX_BUCKETS 17
- #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
- #define CFE_MISSION_ES_DEFAULT_CRC CFE_ES_CrcType_CRC_16
- #define CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)

Checksum/CRC algorithm identifiers

```
• #define CFE_MISSION_ES_CRC_8 CFE_ES_CrcType_CRC_8 /* 1 */
• #define CFE_MISSION_ES_CRC_16 CFE_ES_CrcType_CRC_16 /* 2 */
• #define CFE_MISSION_ES_CRC_32 CFE_ES_CrcType_CRC_32 /* 3 */
• #define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122
• #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')
• #define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768
• #define CFE_MISSION_SB_MAX_PIPES 64
• #define CFE_MISSION_TBL_MAX_NAME_LENGTH 16
• #define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)
• #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 37
• #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_EPOCH_MICROS 0
• #define CFE_MISSION_TIME_FS_FACTOR 789004800
```

12.55.1 Detailed Description

This header file contains the mission configuration parameters and typedefs with mission scope.

This provides values for configurable items that affect the interface(s) of this module. This includes the CMD/T \leftarrow LM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

It is no longer necessary to provide this file directly in the defs directory, but if present, this file is still supported/usable for backward compatibility. To use this file, it should be called "cfe_mission_cfg.h".

Going forward, more fine-grained (module/purposes-specific) header files are included with each submodule. These may be overridden as necessary, but only if a definition within that file needs to be changed from the default. This approach will reduce the amount of duplicate/cloned definitions and better support alternative build configurations in the future.

Note that if this file is present, the fine-grained header files noted above will *not* be used.

12.55.2 Macro Definition Documentation

12.55.2.1 CFE_FS_FILE_CONTENT_ID #define CFE_FS_FILE_CONTENT_ID 0x63464531
Magic Number for cFE compliant files (= 'cFE1')
Definition at line 313 of file example_mission_cfg.h.

12.55.2.2 CFE_FS_HDR_DESC_MAX_LEN #define CFE_FS_HDR_DESC_MAX_LEN 32
Max length of description field in a standard cFE File Header.
Definition at line 311 of file example_mission_cfg.h.

12.55.2.3 CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN #define CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_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.CDSName"

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 262 of file example_mission_cfg.h.

12.55.2.4 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"

This length does not need to include an extra character for NULL termination.

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 228 of file example_mission_cfg.h.

12.55.2.5 CFE_MISSION_ES_CRC_16 #define CFE_MISSION_ES_CRC_16 CFE_ES_CrcType_CRC_16 /* 2 */
Definition at line 270 of file example_mission_cfg.h.

12.55.2.6 CFE_MISSION_ES_CRC_32 #define CFE_MISSION_ES_CRC_32 CFE_ES_CrcType_CRC_32 /* 3 */
Definition at line 271 of file example_mission_cfg.h.

12.55.2.7 CFE_MISSION_ES_CRC_8 #define CFE_MISSION_ES_CRC_8 CFE_ES_CrcType_CRC_8 /* 1 */
Definition at line 269 of file example_mission_cfg.h.

12.55.2.8 CFE_MISSION_ES_DEFAULT_CRC #define CFE_MISSION_ES_DEFAULT_CRC CFE_ES_CrcType_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_ES_CrcType_CRC_16 is supported (see brief in CFE_ES_CrcType_Enum definition in [cfe_es_api_typedefs.h](#))

Definition at line 242 of file example_mission_cfg.h.

12.55.2.9 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 173 of file example_mission_cfg.h.

12.55.2.10 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 190 of file example_mission_cfg.h.

12.55.2.11 CFE_MISSION_ES_POOL_MAX_BUCKETS #define CFE_MISSION_ES_POOL_MAX_BUCKETS 17

Purpose Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS. This definition is used as the array size with the pool stats structure, and therefore should be consistent across all CPUs in a mission, as well as with the ground station.

There is also a platform-specific limit which may be fewer than this value.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

Definition at line 211 of file example_mission_cfg.h.

12.55.2.12 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

This length does not need to include an extra character for NULL termination.

Limits

Not Applicable

Definition at line 297 of file example_mission_cfg.h.

12.55.2.13 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.

This length must include an extra character for NULL termination.

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 125 of file example_mission_cfg.h.

12.55.2.14 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.

This length must include an extra character for NULL termination.

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 99 of file example_mission_cfg.h.

12.55.2.15 CFE_MISSION_MAX_NUM_FILES #define CFE_MISSION_MAX_NUM_FILES 50

Purpose cFE Maximum number of files in a message/data exchange

Description:

The value of this constant dictates the maximum number of files within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS_MAX_NUM_OPEN_FILES 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_NUM_OPEN_FILES 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_NUM_OPEN_FILES 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.

Definition at line 147 of file example_mission_cfg.h.

12.55.2.16 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.

This length must include an extra character for NULL termination.

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 72 of file example_mission_cfg.h.

12.55.2.17 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 definitions 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 357 of file example_mission_cfg.h.

12.55.2.18 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 340 of file example_mission_cfg.h.

12.55.2.19 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 402 of file example_mission_cfg.h.

12.55.2.20 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 characters) of the table name ('TblName') portion of a Full Table Name of the following form: "ApplicationName.TblName"

This length does not need to include an extra character for NULL termination.

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 382 of file example_mission_cfg.h.

12.55.2.21 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 468 of file example_mission_cfg.h.

12.55.2.22 CFE_MISSION_TIME_AT_TONE_WILL_BE #define CFE_MISSION_TIME_AT_TONE_WILL_BE false
Definition at line 469 of file example_mission_cfg.h.

12.55.2.23 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 432 of file example_mission_cfg.h.

12.55.2.24 CFE_MISSION_TIME_CFG_DEFAULT_UTC #define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
Definition at line 433 of file example_mission_cfg.h.

12.55.2.25 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 445 of file example_mission_cfg.h.

12.55.2.26 CFE_MISSION_TIME_DEF_DELAY_SECS #define CFE_MISSION_TIME_DEF_DELAY_SECS 0
Definition at line 527 of file example_mission_cfg.h.

12.55.2.27 CFE_MISSION_TIME_DEF_DELAY_SUBS #define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
Definition at line 528 of file example_mission_cfg.h.

12.55.2.28 CFE_MISSION_TIME_DEF_LEAPS #define CFE_MISSION_TIME_DEF_LEAPS 37
Definition at line 525 of file example_mission_cfg.h.

12.55.2.29 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 ($\text{time} = \text{MET} + \text{STCF} - \text{Leap Seconds}$) 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 519 of file example_mission_cfg.h.

12.55.2.30 CFE_MISSION_TIME_DEF_MET_SUBS #define CFE_MISSION_TIME_DEF_MET_SUBS 0
Definition at line 520 of file example_mission_cfg.h.

12.55.2.31 CFE_MISSION_TIME_DEF_STCF_SECS #define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
Definition at line 522 of file example_mission_cfg.h.

12.55.2.32 CFE_MISSION_TIME_DEF_STCF_SUBS #define CFE_MISSION_TIME_DEF_STCF_SUBS 0
Definition at line 523 of file example_mission_cfg.h.

12.55.2.33 CFE_MISSION_TIME_EPOCH_DAY #define CFE_MISSION_TIME_EPOCH_DAY 1
Definition at line 546 of file example_mission_cfg.h.

12.55.2.34 CFE_MISSION_TIME_EPOCH_HOUR #define CFE_MISSION_TIME_EPOCH_HOUR 0
Definition at line 547 of file example_mission_cfg.h.

12.55.2.35 CFE_MISSION_TIME_EPOCH_MICROS #define CFE_MISSION_TIME_EPOCH_MICROS 0
Definition at line 550 of file example_mission_cfg.h.

12.55.2.36 CFE_MISSION_TIME_EPOCH_MINUTE #define CFE_MISSION_TIME_EPOCH_MINUTE 0
Definition at line 548 of file example_mission_cfg.h.

12.55.2.37 CFE_MISSION_TIME_EPOCH_SECOND #define CFE_MISSION_TIME_EPOCH_SECOND 0
Definition at line 549 of file example_mission_cfg.h.

12.55.2.38 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
Micros - 0 to 999999

Definition at line 545 of file example_mission_cfg.h.

12.55.2.39 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 588 of file example_mission_cfg.h.

12.55.2.40 CFE_MISSION_TIME_MAX_ELAPSED #define CFE_MISSION_TIME_MAX_ELAPSED 200000

Definition at line 494 of file example_mission_cfg.h.

12.55.2.41 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 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 493 of file example_mission_cfg.h.

12.56 cfe/cmake/sample_defs/example_platform_cfg.h File Reference

Macros

- #define CFE_PLATFORM_ENDIAN CCSDS_LITTLE_ENDIAN
- #define CFE_PLATFORM_CORE_MAX_STARTUP_MSEC 30000
- #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_ES_NONVOL_DISK_MOUNT_STRING "/cf"
- #define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"
- #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 256
- #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_CDS_SIZE (128 * 1024)
- #define CFE_PLATFORM_ES_USER_RESERVED_SIZE (1024 * 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_APP_LOG_FILE "/ram/cfe_es_app_info.log"
- #define CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE "/ram/cfe_es_taskinfo.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_POR_SYSLOG_MODE 0
- #define CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE 1
- #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_CDS_MAX_NUM_ENTRIES 512
- #define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
- #define CFE_PLATFORM_ES_POOL_MAX_BUCKETS 17
- #define CFE_PLATFORM_ES_MAX_MEMORY_POOLS 10
- #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_ES_STARTUP_SYNC_POLL_MSEC 50
- #define CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC 1000
- #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_EVS_MAX_EVENT_FILTERS 8
- #define CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST 32
- #define CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC 15
- #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_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_HIGHEST_VALID_MSGID 0x1FFF
- #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 + 128)
• #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_TBL_START_TASK_PRIORITY 70
• #define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
• #define CFE_PLATFORM_TBL_BUFS_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) ((uint32)(_C1) << 24 | (uint32)(_C2)
<< 16 | (uint32)(_C3) << 8 | (uint32)(_C4))
• #define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)
• #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 (1)
• #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_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_TIME_START_TASK_PRIORITY 60
• #define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
• #define CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY 25
• #define CFE_PLATFORM_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
• #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
• #define CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE 8192
```

12.56.1 Detailed Description

This header file contains the internal configuration parameters and typedefs with platform scope. This provides default values for configurable items that do NOT affect the interface(s) of this module. This includes internal parameters, path names, and limit value(s) that are relevant for a specific platform.

Note

It is no longer necessary to provide this file directly in the defs directory, but if present, this file is still supported/usable for backward compatibility. To use this file, it should be called "cfe_platform_cfg.h".

Going forward, more fine-grained (module/purposes-specific) header files are included with each submodule. These may be overridden as necessary, but only if a definition within that file needs to be changed from the default. This approach will reduce the amount of duplicate/cloned definitions and better support alternative build configurations in the future.

Note that if this file is present, the fine-grained header files noted above will *not* be used.

12.56.2 Macro Definition Documentation

12.56.2.1 CFE_PLATFORM_CORE_MAX_STARTUP_MSEC `#define CFE_PLATFORM_CORE_MAX_STARTUP_MS←
EC 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 allotted 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 84 of file example_platform_cfg.h.

12.56.2.2 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 60 of file example_platform_cfg.h.

12.56.2.3 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 responding 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 parameter. Units are number of [CFE_PLATFORM_ES_APP_SCAN_RATE](#) cycles.

Definition at line 288 of file example_platform_cfg.h.

12.56.2.4 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 parameter. millisecond units.

Definition at line 259 of file example_platform_cfg.h.

12.56.2.5 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE #define CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE←
ZE 80000

Definition at line 830 of file example_platform_cfg.h.

12.56.2.6 CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES #define CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES←
ES_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 720 of file example_platform_cfg.h.

12.56.2.7 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 814 of file example_platform_cfg.h.

12.56.2.8 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 16

Definition at line 815 of file example_platform_cfg.h.

12.56.2.9 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 32

Definition at line 816 of file example_platform_cfg.h.

12.56.2.10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 48

Definition at line 817 of file example_platform_cfg.h.

12.56.2.11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 64

Definition at line 818 of file example_platform_cfg.h.

12.56.2.12 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 96

Definition at line 819 of file example_platform_cfg.h.

12.56.2.13 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 128

Definition at line 820 of file example_platform_cfg.h.

12.56.2.14 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 160

Definition at line 821 of file example_platform_cfg.h.

12.56.2.15 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 256

Definition at line 822 of file example_platform_cfg.h.

12.56.2.16 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 512

Definition at line 823 of file example_platform_cfg.h.

12.56.2.17 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 1024

Definition at line 824 of file example_platform_cfg.h.

12.56.2.18 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 2048

Definition at line 825 of file example_platform_cfg.h.

12.56.2.19 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 4096

Definition at line 826 of file example_platform_cfg.h.

12.56.2.20 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 8192

Definition at line 827 of file example_platform_cfg.h.

12.56.2.21 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 16384

Definition at line 828 of file example_platform_cfg.h.

12.56.2.22 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 32768

Definition at line 829 of file example_platform_cfg.h.

12.56.2.23 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 parameter.

Definition at line 365 of file example_platform_cfg.h.

12.56.2.24 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 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 447 of file example_platform_cfg.h.

12.56.2.25 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 521 of file example_platform_cfg.h.

```
12.56.2.26 CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_ER_LOG_FI←  
LE "/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 493 of file example_platform_cfg.h.

```
12.56.2.27 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 507 of file example_platform_cfg.h.

```
12.56.2.28 CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE #define CFE_PLATFORM_ES_DEFAULT_POR←  
_SYSLOG_MODE 0
```

Purpose Define Default System Log Mode following Power On Reset

Description:

Defines the default mode for the operation of the ES System log following a power on reset. 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. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 539 of file example_platform_cfg.h.

12.56.2.29 CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE #define CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE 1

Purpose Define Default System Log Mode following Processor Reset

Description:

Defines the default mode for the operation of the ES System log following a processor reset. 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. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 557 of file example_platform_cfg.h.

12.56.2.30 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 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 707 of file example_platform_cfg.h.

12.56.2.31 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 478 of file example_platform_cfg.h.

```
12.56.2.32 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE "/ram/cfe_es_taskinfo.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 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 462 of file example_platform_cfg.h.

```
12.56.2.33 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 186 of file example_platform_cfg.h.

```
12.56.2.34 CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE #define CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE 256
```

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 200 of file example_platform_cfg.h.

12.56.2.35 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 159 of file example_platform_cfg.h.

12.56.2.36 CFE_PLATFORM_ES_MAX_BLOCK_SIZE #define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000

Definition at line 803 of file example_platform_cfg.h.

12.56.2.37 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 240 of file example_platform_cfg.h.

12.56.2.38 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 173 of file example_platform_cfg.h.

12.56.2.39 CFE_PLATFORM_ES_MAX_MEMORY_POOLS #define CFE_PLATFORM_ES_MAX_MEMORY_POOLS 10

Purpose Maximum number of memory pools

Description:

The upper limit for the number of memory pools that can concurrently exist within the system.

The CFE_SB and CFE_TBL core subsystems each define a memory pool. Individual applications may also create memory pools, so this value should be set sufficiently high enough to support the applications being used on this platform.

Limits:

Must be at least 2 to support CFE core - SB and TBL pools. No specific upper limit.

Definition at line 768 of file example_platform_cfg.h.

12.56.2.40 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 735 of file example_platform_cfg.h.

12.56.2.41 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.

Definition at line 787 of file example_platform_cfg.h.

12.56.2.42 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 16
Definition at line 788 of file example_platform_cfg.h.

12.56.2.43 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 32
Definition at line 789 of file example_platform_cfg.h.

12.56.2.44 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 48
Definition at line 790 of file example_platform_cfg.h.

12.56.2.45 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 64
Definition at line 791 of file example_platform_cfg.h.

12.56.2.46 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 96
Definition at line 792 of file example_platform_cfg.h.

12.56.2.47 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 128
Definition at line 793 of file example_platform_cfg.h.

12.56.2.48 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 160
Definition at line 794 of file example_platform_cfg.h.

12.56.2.49 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 256
Definition at line 795 of file example_platform_cfg.h.

12.56.2.50 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 512
Definition at line 796 of file example_platform_cfg.h.

12.56.2.51 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 1024
Definition at line 797 of file example_platform_cfg.h.

12.56.2.52 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 2048
Definition at line 798 of file example_platform_cfg.h.

12.56.2.53 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 4096
Definition at line 799 of file example_platform_cfg.h.

12.56.2.54 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 8192

Definition at line 800 of file example_platform_cfg.h.

12.56.2.55 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 16384

Definition at line 801 of file example_platform_cfg.h.

12.56.2.56 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 32768

Definition at line 802 of file example_platform_cfg.h.

12.56.2.57 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 404 of file example_platform_cfg.h.

12.56.2.58 CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING #define CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING "/cf"

Purpose Default virtual path for persistent storage

Description:

This configures the default location in the virtual file system for persistent/non-volatile storage. Files such as the startup script, app/library dynamic modules, and configuration tables are expected to be stored in this directory.

Definition at line 127 of file example_platform_cfg.h.

12.56.2.59 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 418 of file example_platform_cfg.h.

12.56.2.60 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 229 of file example_platform_cfg.h.

12.56.2.61 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 Performance 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 681 of file example_platform_cfg.h.

12.56.2.62 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 spawned 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 652 of file example_platform_cfg.h.

```
12.56.2.63 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 spawned 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 666 of file example_platform_cfg.h.

```
12.56.2.64 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 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 573 of file example_platform_cfg.h.

```
12.56.2.65 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 performance analyzer entries the Performance Analyzer Child Task will write to the file between delays.

Definition at line 691 of file example_platform_cfg.h.

```
12.56.2.66 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 593 of file example_platform_cfg.h.

12.56.2.67 CFE_PLATFORM_ES_PERF_FILTMASK_INIT #define CFE_PLATFORM_ES_PERF_FILTMASK_INIT
IT 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 604 of file example_platform_cfg.h.

12.56.2.68 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 583 of file example_platform_cfg.h.

12.56.2.69 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL #define CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
LL ~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 626 of file example_platform_cfg.h.

12.56.2.70 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT #define CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
IT 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 637 of file example_platform_cfg.h.

12.56.2.71 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 615 of file example_platform_cfg.h.

12.56.2.72 CFE_PLATFORM_ES_POOL_MAX_BUCKETS #define CFE_PLATFORM_ES_POOL_MAX_BUCKETS 17

Purpose Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

The ES and CDS block size lists must correlate with this value

Definition at line 750 of file example_platform_cfg.h.

12.56.2.73 CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING #define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"

Purpose Default virtual path for volatile storage

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 143 of file example_platform_cfg.h.

12.56.2.74 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 324 of file example_platform_cfg.h.

12.56.2.75 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 parameter. Units are percentage. A setting of zero will turn this feature off.

Definition at line 348 of file example_platform_cfg.h.

12.56.2.76 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 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 RAM disk sector size is system dependent and should be verified.

Definition at line 306 of file example_platform_cfg.h.

12.56.2.77 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 100 of file example_platform_cfg.h.

12.56.2.78 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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 115 of file example_platform_cfg.h.

12.56.2.79 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 allotted 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 870 of file example_platform_cfg.h.

12.56.2.80 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 852 of file example_platform_cfg.h.

12.56.2.81 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 215 of file example_platform_cfg.h.

12.56.2.82 CFE_PLATFORM_ES_USER_RESERVED_SIZE #define CFE_PLATFORM_ES_USER_RESERVED_SI←
ZE (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 parameter.

Definition at line 385 of file example_platform_cfg.h.

```
12.56.2.83 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 432 of file example_platform_cfg.h.

```
12.56.2.84 CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC #define CFE_PLATFORM_EVS_APP_EVENTS_PER_←  
SEC 15
```

Purpose Sustained number of event messages per second per app before squelching

Description:

Sustained number of events that may be emitted per app per second.

Limits

This number must be less than or equal to [CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST](#). Values lower than 8 may cause functional and unit test failures.

Definition at line 938 of file example_platform_cfg.h.

```
12.56.2.85 CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE #define CFE_PLATFORM_EVS_DEFAULT_APP_DA←  
TA_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 979 of file example_platform_cfg.h.

12.56.2.86 CFE_PLATFORM_EVS_DEFAULT_LOG_FILE #define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
LE "/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 952 of file example_platform_cfg.h.

12.56.2.87 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 1026 of file example_platform_cfg.h.

12.56.2.88 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 1039 of file example_platform_cfg.h.

12.56.2.89 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG #define CFE_PLATFORM_EVS_DEFAULT_TYPE_FL→
AG 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 1010 of file example_platform_cfg.h.

12.56.2.90 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 964 of file example_platform_cfg.h.

12.56.2.91 CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST #define CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST 32

Purpose Maximum number of event before squelching

Description:

Maximum number of events that may be emitted per app per second. Setting this to 0 will cause events to be unrestricted.

Limits

This number must be less than or equal to INT_MAX/1000

Definition at line 926 of file example_platform_cfg.h.

12.56.2.92 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 914 of file example_platform_cfg.h.

12.56.2.93 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 disabled) 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 993 of file example_platform_cfg.h.

12.56.2.94 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 886 of file example_platform_cfg.h.

12.56.2.95 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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 901 of file example_platform_cfg.h.

12.56.2.96 CFE_PLATFORM_SB_BUF_MEMORY_BYTES #define CFE_PLATFORM_SB_BUF_MEMORY_BYT←
ES 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 1134 of file example_platform_cfg.h.

12.56.2.97 CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME #define CFE_PLATFORM_SB_DEFAULT_MAP_FILE←
NAME "/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 1204 of file example_platform_cfg.h.

12.56.2.98 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 1112 of file example_platform_cfg.h.

12.56.2.99 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME #define CFE_PLATFORM_SB_DEFAULT_PIPE_FIL←
ENAME "/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 1187 of file example_platform_cfg.h.

12.56.2.100 CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME #define CFE_PLATFORM_SB_DEFAULT_RO←
UTING_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 1173 of file example_platform_cfg.h.

12.56.2.101 CFE_PLATFORM_SB_FILTER_MASK1 #define CFE_PLATFORM_SB_FILTER_MASK1 [CFE_EVS_FIRST_4_STOP](#)

Definition at line 1222 of file example_platform_cfg.h.

12.56.2.102 CFE_PLATFORM_SB_FILTER_MASK2 #define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
Definition at line 1225 of file example_platform_cfg.h.

12.56.2.103 CFE_PLATFORM_SB_FILTER_MASK3 #define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
Definition at line 1228 of file example_platform_cfg.h.

12.56.2.104 CFE_PLATFORM_SB_FILTER_MASK4 #define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
Definition at line 1231 of file example_platform_cfg.h.

12.56.2.105 CFE_PLATFORM_SB_FILTER_MASK5 #define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
Definition at line 1234 of file example_platform_cfg.h.

12.56.2.106 CFE_PLATFORM_SB_FILTER_MASK6 #define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
Definition at line 1237 of file example_platform_cfg.h.

12.56.2.107 CFE_PLATFORM_SB_FILTER_MASK7 #define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
Definition at line 1240 of file example_platform_cfg.h.

12.56.2.108 CFE_PLATFORM_SB_FILTER_MASK8 #define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
Definition at line 1243 of file example_platform_cfg.h.

12.56.2.109 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 parameters dictates what SB events will be filtered through SB. The filtering will begin after the SB task initializes and stay in effect until a cmd to SB 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 1221 of file example_platform_cfg.h.

12.56.2.110 CFE_PLATFORM_SB_FILTERED_EVENT2 #define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
Definition at line 1224 of file example_platform_cfg.h.

12.56.2.111 CFE_PLATFORM_SB_FILTERED_EVENT3 #define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
Definition at line 1227 of file example_platform_cfg.h.

12.56.2.112 CFE_PLATFORM_SB_FILTERED_EVENT4 #define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
Definition at line 1230 of file example_platform_cfg.h.

12.56.2.113 CFE_PLATFORM_SB_FILTERED_EVENT5 #define CFE_PLATFORM_SB_FILTERED_EVENT5 0
Definition at line 1233 of file example_platform_cfg.h.

12.56.2.114 CFE_PLATFORM_SB_FILTERED_EVENT6 #define CFE_PLATFORM_SB_FILTERED_EVENT6 0
Definition at line 1236 of file example_platform_cfg.h.

12.56.2.115 CFE_PLATFORM_SB_FILTERED_EVENT7 #define CFE_PLATFORM_SB_FILTERED_EVENT7 0
Definition at line 1239 of file example_platform_cfg.h.

12.56.2.116 CFE_PLATFORM_SB_FILTERED_EVENT8 #define CFE_PLATFORM_SB_FILTERED_EVENT8 0
Definition at line 1242 of file example_platform_cfg.h.

12.56.2.117 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 range of valid message ID's, from 0 to CFE_PLATFORM_SB_HIGHEST_VALID_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

Limits

This parameter has a lower limit is 1, and an upper limit of 0xFFFFFFFF.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE_PLATFORM_SB_MAX_MSG_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 1159 of file example_platform_cfg.h.

12.56.2.118 CFE_PLATFORM_SB_MAX_BLOCK_SIZE #define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_M + 128)

Definition at line 1272 of file example_platform_cfg.h.

12.56.2.119 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 1097 of file example_platform_cfg.h.

12.56.2.120 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 effect 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 must be a power of two if software bus message routing hash implementation is being used. Lower than 64 will cause unit test failures, and telemetry reporting is impacted below 32. There is no hard upper limit, but impacts memory footprint. For software bus message routing search implementation the number of msg ids subscribed to impacts performance.

Definition at line 1064 of file example_platform_cfg.h.

12.56.2.121 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 effect 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 1081 of file example_platform_cfg.h.

12.56.2.122 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_PLATFORM_ES_POOL_MAX_BUCKETS](#)

Definition at line 1256 of file example_platform_cfg.h.

12.56.2.123 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16

Definition at line 1257 of file example_platform_cfg.h.

12.56.2.124 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20

Definition at line 1258 of file example_platform_cfg.h.

12.56.2.125 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36

Definition at line 1259 of file example_platform_cfg.h.

12.56.2.126 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64

Definition at line 1260 of file example_platform_cfg.h.

12.56.2.127 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96

Definition at line 1261 of file example_platform_cfg.h.

12.56.2.128 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128

07 128

Definition at line 1262 of file example_platform_cfg.h.

12.56.2.129 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160

08 160

Definition at line 1263 of file example_platform_cfg.h.

12.56.2.130 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256

09 256

Definition at line 1264 of file example_platform_cfg.h.

12.56.2.131 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
10 512

Definition at line 1265 of file example_platform_cfg.h.

12.56.2.132 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
11 1024

Definition at line 1266 of file example_platform_cfg.h.

12.56.2.133 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
12 2048

Definition at line 1267 of file example_platform_cfg.h.

12.56.2.134 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
13 4096

Definition at line 1268 of file example_platform_cfg.h.

12.56.2.135 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
14 8192

Definition at line 1269 of file example_platform_cfg.h.

12.56.2.136 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
15 16384

Definition at line 1270 of file example_platform_cfg.h.

12.56.2.137 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
16 32768

Definition at line 1271 of file example_platform_cfg.h.

12.56.2.138 CFE_PLATFORM_SB_START_TASK_PRIORITY #define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
TY 64

Purpose Define SB Task Priority

Description:

Defines the cFE_SB Task priority.

Limits

Not Applicable

Definition at line 1283 of file example_platform_cfg.h.

12.56.2.139 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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 1298 of file example_platform_cfg.h.

12.56.2.140 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 1345 of file example_platform_cfg.h.

12.56.2.141 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 1459 of file example_platform_cfg.h.

12.56.2.142 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_TABLES.

Definition at line 1400 of file example_platform_cfg.h.

12.56.2.143 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_TBL_BUF_MEMORY_BLOCK_SIZE](#).

Definition at line 1357 of file example_platform_cfg.h.

12.56.2.144 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_PLATFORM_TBL_MAX_NUM_TABLES](#)) and should be set higher if tables are shared between applications.

Definition at line 1413 of file example_platform_cfg.h.

12.56.2.145 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 1386 of file example_platform_cfg.h.

12.56.2.146 CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS #define CFE_PLATFORM_TBL_MAX_NUM_VALIDA←
TIONS 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 1446 of file example_platform_cfg.h.

12.56.2.147 CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS #define CFE_PLATFORM_TBL_MAX_SIMUL←
TANEOUS_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 1428 of file example_platform_cfg.h.

```
12.56.2.148 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_LOADS](#) 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_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS](#) number of tables to fit into [CFE_PLATFORM_TBL_BUF_MEMORY_BYT](#)

Definition at line 1373 of file example_platform_cfg.h.

```
12.56.2.149 CFE_PLATFORM_TBL_START_TASK_PRIORITY #define CFE_PLATFORM_TBL_START_TASK_PRIO←  
RITY 70
```

Purpose Define TBL Task Priority

Description:

Defines the cFE_TBL Task priority.

Limits

Not Applicable

Definition at line 1314 of file example_platform_cfg.h.

```
12.56.2.150 CFE_PLATFORM_TBL_START_TASK_STACK_SIZE #define CFE_PLATFORM_TBL_START_TASK_S←  
TACK_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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 1329 of file example_platform_cfg.h.

```
12.56.2.151 CFE_PLATFORM_TBL_U32FROM4CHARS #define CFE_PLATFORM_TBL_U32FROM4CHARS ( _C1, _C2, _C3, _C4 ) ((uint32) (_C1) << 24 | (uint32) (_C2) << 16 | (uint32) (_C3) << 8 | (uint32) (_C4))
```

Definition at line 1481 of file example_platform_cfg.h.

```
12.56.2.152 CFE_PLATFORM_TBL_VALID_PRID_1 #define CFE_PLATFORM_TBL_VALID_PRID_1 (1)
```

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 1530 of file example_platform_cfg.h.

```
12.56.2.153 CFE_PLATFORM_TBL_VALID_PRID_2 #define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS ('b', 'c', 'd'))
```

Definition at line 1531 of file example_platform_cfg.h.

```
12.56.2.154 CFE_PLATFORM_TBL_VALID_PRID_3 #define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

Definition at line 1532 of file example_platform_cfg.h.

```
12.56.2.155 CFE_PLATFORM_TBL_VALID_PRID_4 #define CFE_PLATFORM_TBL_VALID_PRID_4 0
```

Definition at line 1533 of file example_platform_cfg.h.

```
12.56.2.156 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 1516 of file example_platform_cfg.h.

12.56.2.157 CFE_PLATFORM_TBL_VALID_SCID_1 #define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)

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 1496 of file example_platform_cfg.h.

12.56.2.158 CFE_PLATFORM_TBL_VALID_SCID_2 #define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CH^{'b', 'c', 'd'})

Definition at line 1497 of file example_platform_cfg.h.

12.56.2.159 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 1478 of file example_platform_cfg.h.

12.56.2.160 CFE_PLATFORM_TIME_CFG_CLIENT #define CFE_PLATFORM_TIME_CFG_CLIENT false

Definition at line 1553 of file example_platform_cfg.h.

12.56.2.161 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 dictates the period at which the simulated 'last tone' time is updated. Units are seconds.

Limits

Not Applicable

Definition at line 1710 of file example_platform_cfg.h.

12.56.2.162 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_TIME_CFG_CLIENT AS true. The other must be defined as false.

Definition at line 1552 of file example_platform_cfg.h.

12.56.2.163 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 definition 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_CFG_SIGNAL define to true to enable tone signal commands.

Limits

Not Applicable

Definition at line 1600 of file example_platform_cfg.h.

12.56.2.164 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 1620 of file example_platform_cfg.h.

12.56.2.165 CFE_PLATFORM_TIME_CFG_SRC_GPS #define CFE_PLATFORM_TIME_CFG_SRC_GPS false
Definition at line 1637 of file example_platform_cfg.h.

12.56.2.166 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_CFG_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_TIME_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 1636 of file example_platform_cfg.h.

12.56.2.167 CFE_PLATFORM_TIME_CFG_SRC_TIME #define CFE_PLATFORM_TIME_CFG_SRC_TIME false
Definition at line 1638 of file example_platform_cfg.h.

12.56.2.168 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 1697 of file example_platform_cfg.h.

12.56.2.169 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 1685 of file example_platform_cfg.h.

12.56.2.170 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 1585 of file example_platform_cfg.h.

12.56.2.171 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 1657 of file example_platform_cfg.h.

12.56.2.172 CFE_PLATFORM_TIME_MAX_DELTA_SUBS #define CFE_PLATFORM_TIME_MAX_DELTA_SU←

BS 500000

Definition at line 1658 of file example_platform_cfg.h.

12.56.2.173 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 1670 of file example_platform_cfg.h.

12.56.2.174 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS #define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0

Definition at line 1671 of file example_platform_cfg.h.

12.56.2.175 CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY #define CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY 25

Definition at line 1727 of file example_platform_cfg.h.

12.56.2.176 CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE #define CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE 8192

Definition at line 1746 of file example_platform_cfg.h.

12.56.2.177 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 parameters. Remember that the meaning of each task priority is inverted – a "lower" number has a "higher" priority.

Definition at line 1725 of file example_platform_cfg.h.

12.56.2.178 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 parameters. There are no restrictions on the upper limit however, the maximum stack 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 1744 of file example_platform_cfg.h.

12.56.2.179 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY #define CFE_PLATFORM_TIME_TONE_TASK_PRIO←
RITY 25

Definition at line 1726 of file example_platform_cfg.h.

12.56.2.180 CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE #define CFE_PLATFORM_TIME_TONE_TASK_S←
TACK_SIZE 4096

Definition at line 1745 of file example_platform_cfg.h.

12.57 cfe/cmake/sample_defs/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_SEDMET_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.

12.57.1 Detailed Description

Purpose: This file contains the cFE performance IDs

Design Notes: Each performance id is used to identify something that needs to be measured. Performance ids are limited to the range of 0 to CFE_MISSION_ES_PERF_MAX_IDS - 1. Any performance ids outside of this range will be ignored and will be flagged as an error. Note that performance ids 0-31 are reserved for the cFE Core.

References:

12.57.2 Macro Definition Documentation

12.57.2.1 CFE_MISSION_ES_MAIN_PERF_ID #define CFE_MISSION_ES_MAIN_PERF_ID 1

Performance ID for Executive Services Task.

Definition at line 42 of file sample_perfids.h.

12.57.2.2 CFE_MISSION_ES_PERF_EXIT_BIT #define CFE_MISSION_ES_PERF_EXIT_BIT 31

bit (31) is reserved by the perf utilities

Definition at line 38 of file sample_perfids.h.

12.57.2.3 CFE_MISSION_EVS_MAIN_PERF_ID #define CFE_MISSION_EVS_MAIN_PERF_ID 2

Performance ID for Events Services Task.

Definition at line 43 of file sample_perfids.h.

12.57.2.4 CFE_MISSION_SB_MAIN_PERF_ID #define CFE_MISSION_SB_MAIN_PERF_ID 4

Performance ID for Software Bus Services Task.

Definition at line 45 of file sample_perfids.h.

12.57.2.5 CFE_MISSION_SB_MSG_LIM_PERF_ID #define CFE_MISSION_SB_MSG_LIM_PERF_ID 5

Performance ID for Software Bus Msg Limit Errors.

Definition at line 46 of file sample_perfids.h.

12.57.2.6 CFE_MISSION_SB_PIPE_OFLOW_PERF_ID #define CFE_MISSION_SB_PIPE_OFLOW_PERF_ID 27

Performance ID for Software Bus Pipe Overflow Errors.

Definition at line 47 of file sample_perfids.h.

12.57.2.7 CFE_MISSION_TBL_MAIN_PERF_ID #define CFE_MISSION_TBL_MAIN_PERF_ID 3

Performance ID for Table Services Task.

Definition at line 44 of file sample_perfids.h.

12.57.2.8 CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID #define CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID 8

Performance ID for 1 Hz Local ISR.

Definition at line 51 of file sample_perfids.h.

12.57.2.9 CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID #define CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID 10

Performance ID for 1 Hz Local Task.

Definition at line 54 of file sample_perfids.h.

12.57.2.10 CFE_MISSION_TIME_MAIN_PERF_ID #define CFE_MISSION_TIME_MAIN_PERF_ID 6

Performance ID for Time Services Task.

Definition at line 49 of file sample_perfids.h.

12.57.2.11 CFE_MISSION_TIME_SENDMET_PERF_ID #define CFE_MISSION_TIME_SENDMET_PERF_ID 9

Performance ID for Time ToneSendMET.

Definition at line 53 of file sample_perfids.h.

12.57.2.12 CFE_MISSION_TIME_TONE1HZISR_PERF_ID #define CFE_MISSION_TIME_TONE1HZISR_PERF_ID 7

Performance ID for 1 Hz Tone ISR.

Definition at line 50 of file sample_perfids.h.

12.57.2.13 CFE_MISSION_TIME_TONE1HZTASK_PERF_ID #define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11

Performance ID for 1 Hz Tone Task.

Definition at line 55 of file sample_perfids.h.

12.58 cfe/docs/src/cfe_api.dox File Reference

12.59 cfe/docs/src/cfe_es.dox File Reference

12.60 cfe/docs/src/cfe_evs.dox File Reference

12.61 cfe/docs/src/cfe_frontpage.dox File Reference

12.62 cfe/docs/src/cfe_glossary.dox File Reference

12.63 cfe/docs/src/cfe_sb.dox File Reference

12.64 cfe/docs/src/cfe_tbl.dox File Reference

12.65 cfe/docs/src/cfe_time.dox File Reference

12.66 cfe/docs/src/cfe_xref.dox File Reference

12.67 cfe/docs/src/cfs_versions.dox File Reference

12.68 cfe/modules/config/fsw/inc/cfe_config_external.h File Reference

```
#include "cfe_config_api_typedefs.h"
```

Functions

- void [CFE_Config_SetupPlatformConfigInfo](#) (void)

12.68.1 Detailed Description

Initialization for CFE configuration registry, external items
These are generated from the build system, they are not directly set

12.68.2 Function Documentation

12.68.2.1 CFE_Config_SetupPlatformConfigInfo() void CFE_Config_SetupPlatformConfigInfo (void)

12.69 cfe/modules/config/fsw/inc/cfe_config_init.h File Reference

```
#include "cfe_config_api_typedefs.h"
#include "cfe_config_ids.h"
```

Functions

- void [CFE_Config_SetupBasicBuildInfo](#) (void)
- int32 [CFE_Config_Init](#) (void)

12.69.1 Detailed Description

Function declarations for items implemented in init.c

12.69.2 Function Documentation

12.69.2.1 CFE_Config_Init() int32 CFE_Config_Init (void)

12.69.2.2 CFE_Config_SetupBasicBuildInfo() void CFE_Config_SetupBasicBuildInfo (void)

12.70 cfe/modules/config/fsw/inc/cfe_config_lookup.h File Reference

```
#include "cfe_config_api_typedefs.h"
#include "cfe_config_table.h"
```

Functions

- [CFE_Config_ValueEntry_t](#) * [CFE_Config_LocateConfigRecordByID](#) ([CFE_ConfigId_t](#) ConfigId)
Gets the value record associated with a config ID.

12.70.1 Detailed Description

Function declarations for items implemented in lookup.c

12.70.2 Function Documentation

12.70.2.1 CFE_Config_LocateConfigRecordByID() `CFE_Config_ValueEntry_t* CFE_Config_LocateConfigRecordByID (CFE_ConfigId_t ConfigId)`
Gets the value record associated with a config ID.

12.71 cfe/modules/config/fsw/inc/cfe_config_nametable.h File Reference

```
#include "cfe_configid_offset.h"
```

Data Structures

- struct `CFE_Config_IdNameEntry`

Typedefs

- typedef struct `CFE_Config_IdNameEntry` `CFE_Config_IdNameEntry_t`

Variables

- const `CFE_Config_IdNameEntry_t CFE_CONFIGID_NAMETABLE []`

12.71.1 Detailed Description

This file contains the CFE configuration registry global data definitions.

12.71.2 Typedef Documentation

12.71.2.1 CFE_Config_IdNameEntry_t `typedef struct CFE_Config_IdNameEntry CFE_Config_IdNameEntry_t`

12.71.3 Variable Documentation

12.71.3.1 CFE_CONFIGID_NAMETABLE `const CFE_Config_IdNameEntry_t CFE_CONFIGID_NAMETABLE []`

12.72 cfe/modules/config/fsw/inc/cfe_config_set.h File Reference

```
#include "cfe_config_api_typedefs.h"
```

Functions

- void `CFE_Config_SetValue (CFE_ConfigId_t ConfigId, uint32 Value)`
- void `CFE_Config_SetObjPointer (CFE_ConfigId_t ConfigId, const void *Ptr)`
- void `CFE_Config_SetString (CFE_ConfigId_t ConfigId, const char *Ptr)`
- void `CFE_Config_SetArrayValue (CFE_ConfigId_t ConfigId, const CFE_Config_ArrayValue_t *ArrayPtr)`

12.72.1 Detailed Description

Function declarations for items implemented in set.c

12.72.2 Function Documentation

12.72.2.1 CFE_Config_SetArrayValue() void CFE_Config_SetArrayValue (
 CFE_ConfigId_t ConfigId,
 const CFE_Config_ArrayValue_t * ArrayPtr)

12.72.2.2 CFE_Config_SetObjPointer() void CFE_Config_SetObjPointer (
 CFE_ConfigId_t ConfigId,
 const void * Ptr)

12.72.2.3 CFE_Config_SetString() void CFE_Config_SetString (
 CFE_ConfigId_t ConfigId,
 const char * Ptr)

12.72.2.4 CFE_Config_SetValue() void CFE_Config_SetValue (
 CFE_ConfigId_t ConfigId,
 uint32 Value)

12.73 cfe/modules/config/fsw/inc/cfe_config_table.h File Reference

```
#include "common_types.h"
#include "cfe_config_ids.h"
```

Data Structures

- union **CFE_Config_ValueBuffer**
- struct **CFE_Config_ValueEntry**

TypeDefs

- typedef enum **CFE_ConfigType** **CFE_ConfigType_t**
- typedef union **CFE_Config_ValueBuffer** **CFE_Config_ValueBuffer_t**
- typedef struct **CFE_Config_ValueEntry** **CFE_Config_ValueEntry_t**

Enumerations

- enum **CFE_ConfigType** {
 CFE_ConfigType_UNDEFINED, **CFE_ConfigType_VALUE**, **CFE_ConfigType_STRING**, **CFE_ConfigType_POINTER**,
 CFE_ConfigType_ARRAY }

12.73.1 Detailed Description

This file contains the CFE configuration registry global data definitions.

12.73.2 Typedef Documentation

12.73.2.1 CFE_Config_ValueBuffer_t `typedef union CFE_Config_ValueBuffer CFE_Config_ValueBuffer_t`

12.73.2.2 CFE_Config_ValueEntry_t `typedef struct CFE_Config_ValueEntry CFE_Config_ValueEntry_t`

12.73.2.3 CFE_ConfigType_t `typedef enum CFE_ConfigType CFE_ConfigType_t`

12.73.3 Enumeration Type Documentation

12.73.3.1 CFE_ConfigType `enum CFE_ConfigType`

Enumerator

<code>CFE_ConfigType_UNDEFINED</code>	
<code>CFE_ConfigType_VALUE</code>	Value is an unsigned int
<code>CFE_ConfigType_STRING</code>	Value is a string pointer
<code>CFE_ConfigType_POINTER</code>	Value is a non-string object pointer
<code>CFE_ConfigType_ARRAY</code>	Value is a combination of length and pointer

Definition at line 34 of file cfe_config_table.h.

12.74 cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h File Reference

Macros

- `#define CFE_CPU1_CMD_MID_BASE 0x1800`
Platform command message ID base offset.
- `#define CFE_CPU1_TLM_MID_BASE 0x0800`
Platform telemetry message ID base offset.
- `#define CFE_GLOBAL_CMD_MID_BASE 0x1860`
"Global" command message ID base offset
- `#define CFE_GLOBAL_TLM_MID_BASE 0x0860`
"Global" telemetry message ID base offset
- `#define CFE_PLATFORM_CMD_TOPICID_TO_MIDV(topic) (CFE_CPU1_CMD_MID_BASE | (topic))`
Convert a command topic ID to a MsgID value.
- `#define CFE_PLATFORM_TLM_TOPICID_TO_MIDV(topic) (CFE_CPU1_TLM_MID_BASE | (topic))`
Convert a telemetry topic ID to a MsgID value.
- `#define CFE_GLOBAL_CMD_TOPICID_TO_MIDV(topic) (CFE_GLOBAL_CMD_MID_BASE | (topic))`
Convert a "global" command topic ID to a MsgID value.
- `#define CFE_GLOBAL_TLM_TOPICID_TO_MIDV(topic) (CFE_GLOBAL_TLM_MID_BASE | (topic))`
Convert a "global" telemetry topic ID to a MsgID value.

12.74.1 Detailed Description

This header file contains the platform-specific base msg ID values and logic to convert a topic ID to a message ID value.

12.74.2 Macro Definition Documentation

12.74.2.1 CFE_CPU1_CMD_MID_BASE #define CFE_CPU1_CMD_MID_BASE 0x1800

Platform command message ID base offset.

Example mechanism for setting default command bits and deconflicting MIDs across multiple platforms in a mission. For any sufficiently complex mission this method is typically replaced by a centralized message ID management scheme.
0x1800 - Nominal value for default message ID implementation (V1). This sets the command field and the secondary header present field. Typical V1 command MID range is 0x1800-1FFF. Additional cpus can deconflict message IDs by incrementing this value to provide sub-allocations (0x1900 for example). 0x0080 - Command bit for MISSION_MSGID_V2 message ID implementation (V2). Although this can be used for the value below due to the relatively small set of MIDs in the framework it will not scale so an alternative method of deconfliction is recommended.

Definition at line 46 of file default_cfe_core_api_base_msgids.h.

12.74.2.2 CFE_CPU1_TLM_MID_BASE #define CFE_CPU1_TLM_MID_BASE 0x0800

Platform telemetry message ID base offset.

0x0800 - Nominal for message ID V1 0x0000 - Potential value for MISSION_MSGID_V2, but limited to a range of 0x0000-0x007F since the command bit is 0x0080. Alternative method of deconfliction is recommended.

See [CFE_CPU1_CMD_MID_BASE](#) for more information

Definition at line 58 of file default_cfe_core_api_base_msgids.h.

12.74.2.3 CFE_GLOBAL_CMD_MID_BASE #define CFE_GLOBAL_CMD_MID_BASE 0x1860

"Global" command message ID base offset

0x1860 - Nominal value for message ID V1 0x00E0 - Potential value for MISSION_MSGID_V2, note command bit is 0x0080. Works in limited cases only, alternative method of deconfliction is recommended. See [CFE_CPU1_CMD_MID_BASE](#) for more information

Definition at line 69 of file default_cfe_core_api_base_msgids.h.

12.74.2.4 CFE_GLOBAL_CMD_TOPICID_TO_MIDV #define CFE_GLOBAL_CMD_TOPICID_TO_MIDV(topic) (CFE_GLOBAL_CMD_MID_BASE | (topic))

Convert a "global" command topic ID to a MsgID value.

A global command is one that is not specific to an individual instance of CFE, but rather intended to be broadcast to all CFE instances at the same time.

This is otherwise identical to [CFE_PLATFORM_CMD_TOPICID_TO_MIDV](#)

Definition at line 116 of file default_cfe_core_api_base_msgids.h.

12.74.2.5 CFE_GLOBAL_TLM_MID_BASE #define CFE_GLOBAL_TLM_MID_BASE 0x0860

"Global" telemetry message ID base offset

0x0860 - Nominal value for message ID V1 0x0060 - Potential value for MISSION_MSGID_V2, note command bit is 0x0080. Works in limited cases only, alternative method of deconfliction is recommended. See [CFE_CPU1_CMD_MID_BASE](#) for more information

Definition at line 80 of file default_cfe_core_api_base_msgids.h.

```
12.74.2.6 CFE_GLOBAL_TLM_TOPICID_TO_MIDV #define CFE_GLOBAL_TLM_TOPICID_TO_MIDV( topic ) (CFE_GLOBAL_TLM_MID_BASE | (topic))
```

Convert a "global" telemetry topic ID to a MsgID value.

A global telemetry is one that is not specific to an individual instance of CFE, but rather intended to be broadcast to all CFE instances at the same time.

This is otherwise identical to [CFE_PLATFORM_TLM_TOPICID_TO_MIDV](#)

Definition at line 126 of file default_cfe_core_api_base_msgids.h.

```
12.74.2.7 CFE_PLATFORM_CMD_TOPICID_TO_MIDV #define CFE_PLATFORM_CMD_TOPICID_TO_MIDV( topic ) (CFE_CPU1_CMD_MID_BASE | (topic))
```

Convert a command topic ID to a MsgID value.

This defines the logic to convert a topic ID value into a message ID value. This operates on integer values and should resolve at compile time such that it can be used in e.g. switch/case statements.

Note

The result of this conversion is a simple integer, thus also needs to go through [CFE_SB_ValueToMsgId\(\)](#) to obtain a properly-typed [CFE_SB_MsgId_t](#) for interacting with SB APIs.

Definition at line 93 of file default_cfe_core_api_base_msgids.h.

```
12.74.2.8 CFE_PLATFORM_TLM_TOPICID_TO_MIDV #define CFE_PLATFORM_TLM_TOPICID_TO_MIDV( topic ) (CFE_CPU1_TLM_MID_BASE | (topic))
```

Convert a telemetry topic ID to a MsgID value.

This defines the logic to convert a topic ID value into a message ID value. This operates on integer values and should resolve at compile time such that it can be used in e.g. switch/case statements.

Note

The result of this conversion is a simple integer, thus also needs to go through [CFE_SB_ValueToMsgId\(\)](#) to obtain a properly-typed [CFE_SB_MsgId_t](#) for interacting with SB APIs.

Definition at line 106 of file default_cfe_core_api_base_msgids.h.

12.75 cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h File Reference

Macros

- #define CFE_MISSION_MAX_PATH_LEN 64
- #define CFE_MISSION_MAX_FILE_LEN 20
- #define CFE_MISSION_MAX_API_LEN 20
- #define CFE_MISSION_MAX_NUM_FILES 50

12.75.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

12.75.2 Macro Definition Documentation

12.75.2.1 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.

This length must include an extra character for NULL termination.

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 108 of file default_cfe_core_api_interface_cfg.h.

12.75.2.2 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.

This length must include an extra character for NULL termination.

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 82 of file default_cfe_core_api_interface_cfg.h.

12.75.2.3 CFE_MISSION_MAX_NUM_FILES #define CFE_MISSION_MAX_NUM_FILES 50

Purpose cFE Maximum number of files in a message/data exchange

Description:

The value of this constant dictates the maximum number of files within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS_MAX_NUM_OPEN_FILES 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_NUM_OPEN_FILES 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_NUM_OPEN_FILES 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.

Definition at line 130 of file default_cfe_core_api_interface_cfg.h.

12.75.2.4 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.

This length must include an extra character for NULL termination.

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 55 of file default_cfe_core_api_interface_cfg.h.

12.76 cfe/modules/core_api/config/default_cfe_mission_cfg.h File Reference

```
#include "cfe_core_api_interface_cfg.h"
#include "cfe_es_mission_cfg.h"
#include "cfe_evs_mission_cfg.h"
#include "cfe_sb_mission_cfg.h"
#include "cfe_tbl_mission_cfg.h"
#include "cfe_time_mission_cfg.h"
#include "cfe_fs_mission_cfg.h"
```

12.76.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

12.77 cfe/modules/core_api/config/default_cfe_msgids.h File Reference

```
#include "cfe_es_msgids.h"
#include "cfe_evs_msgids.h"
#include "cfe_sb_msgids.h"
#include "cfe_tbl_msgids.h"
#include "cfe_time_msgids.h"
```

12.77.1 Detailed Description

Purpose: This header file contains the Message Id's for messages used by the cFE core.

12.78 cfe/modules/core_api/fsw/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_msg.h"
#include "cfe_resourceid.h"
#include "cfe_psp.h"
```

12.78.1 Detailed Description

Purpose: cFE header file

Author: David Kobe, the Hammers Company, Inc.

Notes: This header file centralizes the includes for all cFE Applications. It includes all header files necessary to completely define the cFE interface.

12.79 cfe/modules/core_api/fsw/inc/cfe_config.h File Reference

```
#include "common_types.h"
#include "cfe_config_api_typedefs.h"
#include "cfe_config_ids.h"
```

Functions

- [uint32 CFE_Config_GetValue \(CFE_ConfigId_t ConfigId\)](#)
Obtain an integer value correlating to an CFE configuration ID.
- [const void * CFE_Config_GetObjPointer \(CFE_ConfigId_t ConfigId\)](#)
Obtain a pointer value correlating to an CFE configuration ID.
- [CFE_Config_ArrayValue_t CFE_Config_GetArrayValue \(CFE_ConfigId_t ConfigId\)](#)
Obtain an array correlating to an CFE configuration ID.
- [const char * CFE_Config_GetString \(CFE_ConfigId_t ConfigId\)](#)
Obtain a string value correlating to an CFE configuration ID.

- const char * [CFE_Config_GetName](#) (CFE_ConfigId_t ConfigId)
Obtain the name of a CFE configuration ID.
- CFE_ConfigId_t [CFE_Config_GetIdByName](#) (const char *Name)
Obtain the ID value associated with a configuration name.
- void [CFE_Config_IterateAll](#) (void *Arg, CFE_Config_Callback_t Callback)
Iterate all known name/ID value pairs.
- void [CFE_Config_GetVersionString](#) (char *Buf, size_t Size, const char *Component, const char *SrcVersion, const char *CodeName, const char *LastOffcRel)
Obtain the version string for a cFS component or app.

12.79.1 Detailed Description

Title: cFE Status Code Definition Header File

Purpose: Common source of cFE API return status codes.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.79.2 Function Documentation

12.79.2.1 [CFE_Config_GetArrayValue\(\)](#) CFE_Config_ArrayValue_t [CFE_Config_GetArrayValue](#) (CFE_ConfigId_t ConfigId)

Obtain an array correlating to an CFE configuration ID.

Retrieves the CFE_Config_ArrayValue_t value associated with the specified key. This combines an array length (number of elements) and a pointer to the first element.

If no value has been set, or the key is not valid, this returns 0 / NULL.

Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

Returns

Value associated with key

Return values

NULL	if key is not defined or not set
------	----------------------------------

12.79.2.2 [CFE_Config_GetIdByName\(\)](#) CFE_ConfigId_t [CFE_Config_GetIdByName](#) (const char * Name)

Obtain the ID value associated with a configuration name.

Parameters

in	<i>Name</i>	The name of the ID to look up
----	-------------	-------------------------------

Returns

ID associated with name

Return values

<i>CFE_CONFIGID_UNDEFINED</i>	if the name did not correspond to a key
-------------------------------	---

12.79.2.3 CFE_Config_GetName() `const char* CFE_Config_GetName (CFE_ConfigId_t ConfigId)`

Obtain the name of a CFE configuration ID.

Retrieves the printable name associated with the specified key.

Note

This function does not return NULL.

If the ID is not valid/known, then the implementation returns the special string '[unknown]' rather than NULL, so this function may be more easily used in printf() style calls.

Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

Returns

Name associated with key

12.79.2.4 CFE_Config_GetObjPointer() `const void* CFE_Config_GetObjPointer (CFE_ConfigId_t ConfigId)`

Obtain a pointer value correlating to an CFE configuration ID.

Retrieves the pointer value associated with the specified key.

If no value has been set, or the key is not valid, this returns NULL.

Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

Returns

Value associated with key

Return values

<i>NULL</i>	if key is not defined or not set
-------------	----------------------------------

12.79.2.5 CFE_Config_GetString() `const char* CFE_Config_GetString (`
`CFE_ConfigId_t ConfigId)`

Obtain a string value correlating to an CFE configuration ID.
Retrieves the string value associated with the specified key.
If no value has been set, or the key is not valid, this returns the special string "UNDEFINED"

Note

This function does not return NULL, so it can be used directly in printf-style calls.

Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

Returns

String value associated with key

12.79.2.6 CFE_Config_GetValue() `uint32 CFE_Config_GetValue (`
`CFE_ConfigId_t ConfigId)`

Obtain an integer value correlating to an CFE configuration ID.
Retrieves the integer value associated with the specified key.
If no value has been set, or the key is not valid, this returns 0.

Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

Returns

Value associated with key

Return values

0	if key is not defined or not set
---	----------------------------------

12.79.2.7 CFE_Config_GetVersionString() `void CFE_Config_GetVersionString (`
`char * Buf,`
`size_t Size,`
`const char * Component,`
`const char * SrcVersion,`
`const char * CodeName,`
`const char * LastOffcRel)`

Obtain the version string for a cFS component or app.
Assembles a standardized version string associated with the specified component/app.

Parameters

in	<i>Buf</i>	Buffer to place version string in. Will be populated with standard version string containing the provided parameters (i.e.: "cFE DEVELOPMENT BUILD equuleus-rc1+dev0 (Codename equuleus), Last Official Release: cFE 6.7.0"
in	<i>Size</i>	Size of the provided buffer
in	<i>Component</i>	Component for which to get version string (i.e. "cFE")
in	<i>SrcVersion</i>	Source version identifier (i.e. "equuleus-rc1+dev0")
in	<i>CodeName</i>	Code name for the build (i.e. "equuleus")
in	<i>LastOffcRel</i>	Last official release (i.e. "6.7.0")

```
12.79.2.8 CFE_Config_IterateAll() void CFE_Config_IterateAll (
    void * Arg,
    CFE_Config_Callback_t Callback )
```

Iterate all known name/ID value pairs.

Parameters

in	<i>Arg</i>	User-supplied opaque argument to pass to callback
in	<i>Callback</i>	User-supplied callback function to invoke for each ID

12.80 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_api_typedefs.h"
```

Data Structures

- struct [CFE_Config_ArrayValue](#)

Wrapper type for array configuration.

Macros

- #define [CFE_CONFIGID_C](#)(val) (([CFE_ConfigId_t](#))CFE_RESOURCEID_WRAP(val))
- #define [CFE_CONFIGID_UNDEFINED](#) [CFE_CONFIGID_C](#)(CFE_RESOURCEID_UNDEFINED)

Typedefs

- typedef CFE_RESOURCEID_BASE_TYPE [CFE_ConfigId_t](#)

A type for Configuration IDs.

- typedef void(* [CFE_Config_Callback_t](#)) (void *Arg, [CFE_ConfigId_t](#) Id, const char *Name)
- typedef struct [CFE_Config_ArrayValue](#) [CFE_Config_ArrayValue_t](#)

Wrapper type for array configuration.

12.80.1 Detailed Description

Title: cFE Status Code Definition Header File

Purpose: Common source of cFE API return status codes.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.80.2 Macro Definition Documentation

12.80.2.1 CFE_CONFIGID_C `#define CFE_CONFIGID_C(`
`val) ((CFE_ConfigId_t)CFE_RESOURCEID_WRAP(val))`

Definition at line 48 of file cfe_config_api_typedefs.h.

12.80.2.2 CFE_CONFIGID_UNDEFINED `#define CFE_CONFIGID_UNDEFINED CFE_CONFIGID_C(CFE_RESOURCEID_UNDEFINED)`
Definition at line 49 of file cfe_config_api_typedefs.h.

12.80.3 Typedef Documentation

12.80.3.1 CFE_Config_ArrayValue_t `typedef struct CFE_Config_ArrayValue CFE_Config_ArrayValue_t`
Wrapper type for array configuration.

This is a pair containing a size and pointer that is get/set via a single config table entry

12.80.3.2 CFE_Config_Callback_t `typedef void(* CFE_Config_Callback_t)(void *Arg, CFE_ConfigId_t`
`Id, const char *Name)`

Definition at line 51 of file cfe_config_api_typedefs.h.

12.80.3.3 CFE_ConfigId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ConfigId_t`
A type for Configuration IDs.

This is the type that is used for any API accepting or returning a configuration key ID

Definition at line 46 of file cfe_config_api_typedefs.h.

12.81 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference

```
#include "common_types.h"
```

Macros

- `#define CFE_MAKE_BIG16(n) (((n)&0x00FF) << 8) | (((n)&0xFF00) >> 8))`
- `#define CFE_MAKE_BIG32(n) (((n)&0x000000FF) << 24) | (((n)&0x0000FF00) << 8) | (((n)&0x00FF0000) >> 8) | (((n)&0xFF000000) >> 24))`

12.81.1 Detailed Description

Purpose: Define macros to enforce big-endian/network byte order for 16 and 32 bit integers

12.81.2 Macro Definition Documentation

12.81.2.1 CFE_MAKE_BIG16 `#define CFE_MAKE_BIG16(n) (((n)&0x00FF) << 8) | (((n)&0xFF00) >> 8))`

Definition at line 64 of file cfe_endian.h.

12.81.2.2 CFE_MAKE_BIG32 `#define CFE_MAKE_BIG32(n) (((((n)&0x000000FF) << 24) | (((n)&0x0000FF00) << 8) | (((n)&0x00FF0000) >> 8) | (((n)&0xFF000000) >> 24))`

Definition at line 65 of file cfe_endian.h.

12.82 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference

```
#include "osapi.h"
```

Macros

- `#define CFE_STATUS_C(X) ((CFE_Status_t)(X))`
cFE Status macro for literal
- `#define CFE_STATUS_STRING_LENGTH 11`
cFE Status converted to string length limit
- `#define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)`
Error Severity Bitmask.
- `#define CFE_SEVERITY_SUCCESS ((CFE_Status_t)0x00000000)`
Severity Success.
- `#define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)`
Severity Info.
- `#define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)`
Severity Error.
- `#define CFE_SERVICE_BITMASK ((CFE_Status_t)0x0e000000)`
Error Service Bitmask.
- `#define CFE_EVENTS_SERVICE ((CFE_Status_t)0x02000000)`
Event Service.
- `#define CFE_EXECUTIVE_SERVICE ((CFE_Status_t)0x04000000)`
Executive Service.
- `#define CFE_FILE_SERVICE ((CFE_Status_t)0x06000000)`
File Service.
- `#define CFE_GENERIC_SERVICE ((CFE_Status_t)0x08000000)`
Generic Service.
- `#define CFE_SOFTWARE_BUS_SERVICE ((CFE_Status_t)0x0a000000)`
Software Bus Service.
- `#define CFE_TABLE_SERVICE ((CFE_Status_t)0x0c000000)`
Table Service.
- `#define CFE_TIME_SERVICE ((CFE_Status_t)0x0e000000)`
Time Service.
- `#define CFE_SUCCESS ((CFE_Status_t)0)`

- Successful execution.*
- #define CFE_STATUS_NO_COUNTER_INCREMENT ((CFE_Status_t)0x48000001)
No Counter Increment.
 - #define CFE_STATUS_WRONG_MSG_LENGTH ((CFE_Status_t)0xc8000002)
Wrong Message Length.
 - #define CFE_STATUS_UNKNOWN_MSG_ID ((CFE_Status_t)0xc8000003)
Unknown Message ID.
 - #define CFE_STATUS_BAD_COMMAND_CODE ((CFE_Status_t)0xc8000004)
Bad Command Code.
 - #define CFE_STATUS_EXTERNAL_RESOURCE_FAIL ((CFE_Status_t)0xc8000005)
External failure.
 - #define CFE_STATUS_REQUEST_ALREADY_PENDING ((int32)0xc8000006)
Request already pending.
 - #define CFE_STATUS_VALIDATION_FAILURE ((int32)0xc8000007)
Request or input value failed basic structural validation.
 - #define CFE_STATUS_RANGE_ERROR ((int32)0xc8000008)
Request or input value is out of range.
 - #define CFE_STATUS_INCORRECT_STATE ((int32)0xc8000009)
Cannot process request at this time.
 - #define CFE_STATUS_NOT_IMPLEMENTED ((CFE_Status_t)0xc800ffff)
Not Implemented.
 - #define CFE_EVS_UNKNOWN_FILTER ((CFE_Status_t)0xc2000001)
Unknown Filter.
 - #define CFE_EVS_APP_NOT_REGISTERED ((CFE_Status_t)0xc2000002)
Application Not Registered.
 - #define CFE_EVS_APP_ILLEGAL_APP_ID ((CFE_Status_t)0xc2000003)
Illegal Application ID.
 - #define CFE_EVS_APP_FILTER_OVERLOAD ((CFE_Status_t)0xc2000004)
Application Filter Overload.
 - #define CFE_EVS_RESET_AREA_POINTER ((CFE_Status_t)0xc2000005)
Reset Area Pointer Failure.
 - #define CFE_EVS_EVT_NOT_REGISTERED ((CFE_Status_t)0xc2000006)
Event Not Registered.
 - #define CFE_EVS_FILE_WRITE_ERROR ((CFE_Status_t)0xc2000007)
File Write Error.
 - #define CFE_EVS_INVALID_PARAMETER ((CFE_Status_t)0xc2000008)
Invalid Pointer.
 - #define CFE_EVS_APP_SQUELCHED ((CFE_Status_t)0xc2000009)
Event squelched.
 - #define CFE_EVS_NOT_IMPLEMENTED ((CFE_Status_t)0xc200ffff)
Not Implemented.
 - #define CFE_ES_ERR_RESOURCEID_NOT_VALID ((CFE_Status_t)0xc4000001)
Resource ID is not valid.
 - #define CFE_ES_ERR_NAME_NOT_FOUND ((CFE_Status_t)0xc4000002)
Resource Name Error.
 - #define CFE_ES_ERR_APP_CREATE ((CFE_Status_t)0xc4000004)
Application Create Error.

- #define CFE_ES_ERR_CHILD_TASK_CREATE ((CFE_Status_t)0xc4000005)
Child Task Create Error.
- #define CFE_ES_ERR_SYS_LOG_FULL ((CFE_Status_t)0xc4000006)
System Log Full.
- #define CFE_ES_ERR_MEM_BLOCK_SIZE ((CFE_Status_t)0xc4000008)
Memory Block Size Error.
- #define CFE_ES_ERR_LOAD_LIB ((CFE_Status_t)0xc4000009)
Load Library Error.
- #define CFE_ES_BAD_ARGUMENT ((CFE_Status_t)0xc400000a)
Bad Argument.
- #define CFE_ES_ERR_CHILD_TASK_REGISTER ((CFE_Status_t)0xc400000b)
Child Task Register Error.
- #define CFE_ES_CDS_ALREADY_EXISTS ((CFE_Status_t)0x4400000d)
CDS Already Exists.
- #define CFE_ES_CDS_INSUFFICIENT_MEMORY ((CFE_Status_t)0xc400000e)
CDS Insufficient Memory.
- #define CFE_ES_CDS_INVALID_NAME ((CFE_Status_t)0xc400000f)
CDS Invalid Name.
- #define CFE_ES_CDS_INVALID_SIZE ((CFE_Status_t)0xc4000010)
CDS Invalid Size.
- #define CFE_ES_CDS_INVALID ((CFE_Status_t)0xc4000012)
CDS Invalid.
- #define CFE_ES_CDS_ACCESS_ERROR ((CFE_Status_t)0xc4000013)
CDS Access Error.
- #define CFE_ES_FILE_IO_ERR ((CFE_Status_t)0xc4000014)
File IO Error.
- #define CFE_ES_RST_ACCESS_ERR ((CFE_Status_t)0xc4000015)
Reset Area Access Error.
- #define CFE_ES_ERR_APP_REGISTER ((CFE_Status_t)0xc4000017)
Application Register Error.
- #define CFE_ES_ERR_CHILD_TASK_DELETE ((CFE_Status_t)0xc4000018)
Child Task Delete Error.
- #define CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK ((CFE_Status_t)0xc4000019)
Child Task Delete Passed Main Task.
- #define CFE_ES_CDS_BLOCK_CRC_ERR ((CFE_Status_t)0xc400001A)
CDS Block CRC Error.
- #define CFE_ES_MUT_SEM_DELETE_ERR ((CFE_Status_t)0xc400001B)
Mutex Semaphore Delete Error.
- #define CFE_ES_BIN_SEM_DELETE_ERR ((CFE_Status_t)0xc400001C)
Binary Semaphore Delete Error.
- #define CFE_ES_COUNT_SEM_DELETE_ERR ((CFE_Status_t)0xc400001D)
Counting Semaphore Delete Error.
- #define CFE_ES_QUEUE_DELETE_ERR ((CFE_Status_t)0xc400001E)
Queue Delete Error.
- #define CFE_ES_FILE_CLOSE_ERR ((CFE_Status_t)0xc400001F)
File Close Error.
- #define CFE_ES_CDS_WRONG_TYPE_ERR ((CFE_Status_t)0xc4000020)

- #define CFE_ES_CDS_OWNER_ACTIVE_ERR ((CFE_Status_t)0xc4000022)
CDS Owner Active Error.
- #define CFE_ES_APP_CLEANUP_ERR ((CFE_Status_t)0xc4000023)
Application Cleanup Error.
- #define CFE_ES_TIMER_DELETE_ERR ((CFE_Status_t)0xc4000024)
Timer Delete Error.
- #define CFE_ES_BUFFER_NOT_IN_POOL ((CFE_Status_t)0xc4000025)
Buffer Not In Pool.
- #define CFE_ES_TASK_DELETE_ERR ((CFE_Status_t)0xc4000026)
Task Delete Error.
- #define CFE_ES_OPERATION_TIMED_OUT ((CFE_Status_t)0xc4000027)
Operation Timed Out.
- #define CFE_ES_LIB_ALREADY_LOADED ((CFE_Status_t)0x44000028)
Library Already Loaded.
- #define CFE_ES_ERR_SYS_LOG_TRUNCATED ((CFE_Status_t)0x44000029)
System Log Message Truncated.
- #define CFE_ES_NO_RESOURCE_IDS_AVAILABLE ((CFE_Status_t)0xc400002B)
Resource ID is not available.
- #define CFE_ES_POOL_BLOCK_INVALID ((CFE_Status_t)0xc400002C)
Invalid pool block.
- #define CFE_ES_ERR_DUPLICATE_NAME ((CFE_Status_t)0xc400002E)
Duplicate Name Error.
- #define CFE_ES_NOT_IMPLEMENTED ((CFE_Status_t)0xc400ffff)
Not Implemented.
- #define CFE_FS_BAD_ARGUMENT ((CFE_Status_t)0xc6000001)
Bad Argument.
- #define CFE_FS_INVALID_PATH ((CFE_Status_t)0xc6000002)
Invalid Path.
- #define CFE_FS_FNAME_TOO_LONG ((CFE_Status_t)0xc6000003)
Filename Too Long.
- #define CFE_FS_NOT_IMPLEMENTED ((CFE_Status_t)0xc600ffff)
Not Implemented.
- #define CFE_SB_TIME_OUT ((CFE_Status_t)0xca000001)
Time Out.
- #define CFE_SB_NO_MESSAGE ((CFE_Status_t)0xca000002)
No Message.
- #define CFE_SB_BAD_ARGUMENT ((CFE_Status_t)0xca000003)
Bad Argument.
- #define CFE_SB_MAX_PIPES_MET ((CFE_Status_t)0xca000004)
Max Pipes Met.
- #define CFE_SB_PIPE_CR_ERR ((CFE_Status_t)0xca000005)
Pipe Create Error.
- #define CFE_SB_PIPE_RD_ERR ((CFE_Status_t)0xca000006)
Pipe Read Error.
- #define CFE_SB_MSG_TOO_BIG ((CFE_Status_t)0xca000007)
Message Too Big.

- #define CFE_SB_BUF_ALOC_ERR ((CFE_Status_t)0xca000008)
Buffer Allocation Error.
- #define CFE_SB_MAX_MSGS_MET ((CFE_Status_t)0xca000009)
Max Messages Met.
- #define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca00000a)
Max Destinations Met.
- #define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca00000c)
Internal Error.
- #define CFE_SB_WRONG_MSG_TYPE ((CFE_Status_t)0xca00000d)
Wrong Message Type.
- #define CFE_SB_BUFFER_INVALID ((CFE_Status_t)0xca00000e)
Buffer Invalid.
- #define CFE_SB_NOT_IMPLEMENTED ((CFE_Status_t)0xca00ffff)
Not Implemented.
- #define CFE_TBL_ERR_INVALID_HANDLE ((CFE_Status_t)0xcc000001)
Invalid Handle.
- #define CFE_TBL_ERR_INVALID_NAME ((CFE_Status_t)0xcc000002)
Invalid Name.
- #define CFE_TBL_ERR_INVALID_SIZE ((CFE_Status_t)0xcc000003)
Invalid Size.
- #define CFE_TBL_INFO_UPDATE_PENDING ((CFE_Status_t)0x4c000004)
Update Pending.
- #define CFE_TBL_ERR_NEVER_LOADED ((CFE_Status_t)0xcc000005)
Never Loaded.
- #define CFE_TBL_ERR_REGISTRY_FULL ((CFE_Status_t)0xcc000006)
Registry Full.
- #define CFE_TBL_WARN_DUPLICATE ((CFE_Status_t)0x4c000007)
Duplicate Warning.
- #define CFE_TBL_ERR_NO_ACCESS ((CFE_Status_t)0xcc000008)
No Access.
- #define CFE_TBL_ERR_UNREGISTERED ((CFE_Status_t)0xcc000009)
Unregistered.
- #define CFE_TBL_ERR_HANDLES_FULL ((CFE_Status_t)0xcc00000B)
Handles Full.
- #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((CFE_Status_t)0xcc00000C)
Duplicate Table With Different Size.
- #define CFE_TBL_ERR_DUPLICATE_NOT_OWNED ((CFE_Status_t)0xcc00000D)
Duplicate Table And Not Owned.
- #define CFE_TBL_INFO_UPDATED ((CFE_Status_t)0x4c00000E)
Updated.
- #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((CFE_Status_t)0xcc00000F)
No Buffer Available.
- #define CFE_TBL_ERR_DUMP_ONLY ((CFE_Status_t)0xcc000010)
Dump Only Error.
- #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((CFE_Status_t)0xcc000011)
Illegal Source Type.
- #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((CFE_Status_t)0xcc000012)

Load In Progress.

- #define CFE_TBL_ERR_FILE_TOO_LARGE ((CFE_Status_t)0xcc000014)
File Too Large.
- #define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)
Short File Warning.
- #define CFE_TBL_ERR_BAD_CONTENT_ID ((CFE_Status_t)0xcc000016)
Bad Content ID.
- #define CFE_TBL_INFO_NO_UPDATE_PENDING ((CFE_Status_t)0x4c000017)
No Update Pending.
- #define CFE_TBL_INFO_TABLE_LOCKED ((CFE_Status_t)0x4c000018)
Table Locked.
- #define CFE_TBL_INFO_VALIDATION_PENDING ((CFE_Status_t)0x4c000019)
- #define CFE_TBL_INFO_NO_VALIDATION_PENDING ((CFE_Status_t)0x4c00001A)
- #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((CFE_Status_t)0xcc00001B)
Bad Subtype ID.
- #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((CFE_Status_t)0xcc00001C)
File Size Inconsistent.
- #define CFE_TBL_ERR_NO_STD_HEADER ((CFE_Status_t)0xcc00001D)
No Standard Header.
- #define CFE_TBL_ERR_NO_TBL_HEADER ((CFE_Status_t)0xcc00001E)
No Table Header.
- #define CFE_TBL_ERR_FILENAME_TOO_LONG ((CFE_Status_t)0xcc00001F)
Filenname Too Long.
- #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc000020)
File For Wrong Table.
- #define CFE_TBL_ERR_LOAD_INCOMPLETE ((CFE_Status_t)0xcc000021)
Load Incomplete.
- #define CFE_TBL_WARN_PARTIAL_LOAD ((CFE_Status_t)0x4c000022)
Partial Load Warning.
- #define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)
Partial Load Error.
- #define CFE_TBL_INFO_DUMP_PENDING ((CFE_Status_t)0x4c000024)
Dump Pending.
- #define CFE_TBL_ERR_INVALID_OPTIONS ((CFE_Status_t)0xcc000025)
Invalid Options.
- #define CFE_TBL_WARN_NOT_CRITICAL ((CFE_Status_t)0x4c000026)
Not Critical Warning.
- #define CFE_TBL_INFO_RECOVERED_TBL ((CFE_Status_t)0x4c000027)
Recovered Table.
- #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((CFE_Status_t)0xcc000028)
Bad Spacecraft ID.
- #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((CFE_Status_t)0xcc000029)
Bad Processor ID.
- #define CFE_TBL_MESSAGE_ERROR ((CFE_Status_t)0xcc00002a)
Message Error.
- #define CFE_TBL_ERR_SHORT_FILE ((CFE_Status_t)0xcc00002b)
- #define CFE_TBL_ERR_ACCESS ((CFE_Status_t)0xcc00002c)

- #define CFE_TBL_BAD_ARGUMENT ((CFE_Status_t)0xcc00002d)
Bad Argument.
- #define CFE_TBL_NOT_IMPLEMENTED ((CFE_Status_t)0xcc00ffff)
Not Implemented.
- #define CFE_TIME_NOT_IMPLEMENTED ((CFE_Status_t)0xce00ffff)
Not Implemented.
- #define CFE_TIME_INTERNAL_ONLY ((CFE_Status_t)0xce000001)
Internal Only.
- #define CFE_TIME_OUT_OF_RANGE ((CFE_Status_t)0xce000002)
Out Of Range.
- #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((CFE_Status_t)0xce000003)
Too Many Sync Callbacks.
- #define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000004)
Callback Not Registered.
- #define CFE_TIME_BAD_ARGUMENT ((CFE_Status_t)0xce000005)
Bad Argument.

Typedefs

- typedef int32 CFE_Status_t
cFE Status type for readability and eventually type safety
- typedef char CFE_StatusString_t[CFE_STATUS_STRING_LENGTH]
For the `CFE_ES_StatusToString()` function, to ensure everyone is making an array of the same length.

Functions

- char * CFE_ES_StatusToString (CFE_Status_t status, CFE_StatusString_t *status_string)
Convert status to a string.

12.82.1 Detailed Description

Title: cFE Status Code Definition Header File

Purpose: Common source of cFE API return status codes.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.82.2 Macro Definition Documentation

12.82.2.1 CFE_EVENTS_SERVICE #define CFE_EVENTS_SERVICE ((CFE_Status_t)0x02000000)
Event Service.
Definition at line 126 of file cfe_error.h.

12.82.2.2 CFE_EXECUTIVE_SERVICE #define CFE_EXECUTIVE_SERVICE ((CFE_Status_t)0x04000000)
Executive Service.
Definition at line 127 of file cfe_error.h.

12.82.2.3 CFE_FILE_SERVICE #define CFE_FILE_SERVICE ((CFE_Status_t)0x06000000)

File Service.

Definition at line 128 of file cfe_error.h.

12.82.2.4 CFE_GENERIC_SERVICE #define CFE_GENERIC_SERVICE ((CFE_Status_t)0x08000000)

Generic Service.

Definition at line 129 of file cfe_error.h.

12.82.2.5 CFE_SERVICE_BITMASK #define CFE_SERVICE_BITMASK ((CFE_Status_t)0x0e000000)

Error Service Bitmask.

Definition at line 124 of file cfe_error.h.

12.82.2.6 CFE_SEVERITY_BITMASK #define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)

Error Severity Bitmask.

Definition at line 115 of file cfe_error.h.

12.82.2.7 CFE_SEVERITY_ERROR #define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)

Severity Error.

Definition at line 119 of file cfe_error.h.

12.82.2.8 CFE_SEVERITY_INFO #define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)

Severity Info.

Definition at line 118 of file cfe_error.h.

12.82.2.9 CFE_SEVERITY_SUCCESS #define CFE_SEVERITY_SUCCESS ((CFE_Status_t)0x00000000)

Severity Success.

Definition at line 117 of file cfe_error.h.

12.82.2.10 CFE_SOFTWARE_BUS_SERVICE #define CFE_SOFTWARE_BUS_SERVICE ((CFE_Status_t)0x0a000000)

Software Bus Service.

Definition at line 130 of file cfe_error.h.

12.82.2.11 CFE_STATUS_C #define CFE_STATUS_C(

X) ((CFE_Status_t)(X))

cFE Status macro for literal

Definition at line 48 of file cfe_error.h.

12.82.2.12 CFE_STATUS_STRING_LENGTH #define CFE_STATUS_STRING_LENGTH 11

cFE Status converted to string length limit

Used for sizing CFE_StatusString_t intended for use in printing CFE_Status_t values Sized for 0x%08x and NULL

Definition at line 56 of file cfe_error.h.

12.82.2.13 CFE_TABLE_SERVICE #define CFE_TABLE_SERVICE ((CFE_Status_t)0x0c000000)
Table Service.
Definition at line 131 of file cfe_error.h.

12.82.2.14 CFE_TIME_SERVICE #define CFE_TIME_SERVICE ((CFE_Status_t)0x0e000000)
Time Service.
Definition at line 132 of file cfe_error.h.

12.82.3 Typedef Documentation

12.82.3.1 CFE_Status_t typedef int32 CFE_Status_t
cFE Status type for readability and eventually type safety
Definition at line 43 of file cfe_error.h.

12.82.3.2 CFE_StatusString_t typedef char CFE_StatusString_t[CFE_STATUS_STRING_LENGTH]
For the [CFE_ES_StatusToString\(\)](#) function, to ensure everyone is making an array of the same length.
Definition at line 62 of file cfe_error.h.

12.82.4 Function Documentation

12.82.4.1 CFE_ES_StatusToString() char* CFE_ES_StatusToString (
 CFE_Status_t status,
 CFE_StatusString_t * status_string)

Convert status to a string.

Parameters

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

Returns

Passed in string pointer

12.83 cfe/modules/core_api/fsw/inc/cfe_es.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_resourceid_api_typedefs.h"
```

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_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.

Functions

- CFE_Status_t CFE_ES_AppID_ToIndex (CFE_ES_AppId_t AppID, uint32 *Idx)
Obtain an index value correlating to an ES Application ID.
- int32 CFE_ES_LibID_ToIndex (CFE_ES_LibId_t LibId, uint32 *Idx)
Obtain an index value correlating to an ES Library ID.
- CFE_Status_t CFE_ES_TaskID_ToIndex (CFE_ES_TaskId_t TaskID, uint32 *Idx)
Obtain an index value correlating to an ES Task ID.
- CFE_Status_t CFE_ES_CounterID_ToIndex (CFE_ES_CounterId_t CounterId, uint32 *Idx)
Obtain an index value correlating to an ES Counter ID.
- void CFE_ES_Main (uint32 StartType, uint32 StartSubtype, uint32 ModelId, const char *StartFilePath)
cFE Main Entry Point used by Board Support Package to start cFE
- CFE_Status_t CFE_ES_ResetCFE (uint32 ResetType)
Reset the cFE Core and all cFE Applications.
- CFE_Status_t CFE_ES_RestartApp (CFE_ES_AppId_t AppID)
Restart a single cFE Application.
- CFE_Status_t CFE_ES_ReloadApp (CFE_ES_AppId_t AppID, const char *AppFileName)
Reload a single cFE Application.
- CFE_Status_t CFE_ES_DeleteApp (CFE_ES_AppId_t 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.
- CFE_Status_t 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.
- void CFE_ES_IncrementTaskCounter (void)
Increments the execution counter for the calling task.
- int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)
Return the most recent Reset Type.
- CFE_Status_t CFE_ES_GetAppID (CFE_ES_AppId_t *AppIdPtr)
Get an Application ID for the calling Application.
- CFE_Status_t CFE_ES_GetTaskID (CFE_ES_TaskId_t *TaskIdPtr)
Get the task ID of the calling context.
- CFE_Status_t CFE_ES_GetAppIDByName (CFE_ES_AppId_t *AppIdPtr, const char *AppName)
Get an Application ID associated with a specified Application name.
- CFE_Status_t CFE_ES_GetLibIDByName (CFE_ES_LibId_t *LibIdPtr, const char *LibName)
Get a Library ID associated with a specified Library name.
- CFE_Status_t CFE_ES_GetAppName (char *AppName, CFE_ES_AppId_t AppId, size_t BufferLength)

- `CFE_Status_t CFE_ES_GetLibName (char *LibName, CFE_ES_LibId_t LibId, size_t BufferLength)`
Get an Application name for a specified Application ID.
- `CFE_Status_t CFE_ES_GetApplInfo (CFE_ES_AppInfo_t *ApplInfo, CFE_ES_AppId_t AppId)`
Get a Library name for a specified Library ID.
- `CFE_Status_t CFE_ES_GetTaskInfo (CFE_ES_TaskInfo_t *TaskInfo, CFE_ES_TaskId_t TaskId)`
Get Application Information given a specified App ID.
- `int32 CFE_ES_GetLibInfo (CFE_ES_AppInfo_t *LibInfo, CFE_ES_LibId_t LibId)`
Get Task Information given a specified Task ID.
- `int32 CFE_ES_GetModuleInfo (CFE_ES_AppInfo_t *ModuleInfo, CFE_Resourceld_t Resourceld)`
Get Library Information given a specified Resource ID.
- `int32 CFE_ES_CreateChildTask (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr FunctionPtr, CFE_ES_StackPointer_t StackPtr, size_t StackSize, CFE_ES_TaskPriority_Atom_t Priority, uint32 Flags)`
Get Information given a specified Resource ID.
- `CFE_Status_t CFE_ES_CreateChildTask (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr FunctionPtr, CFE_ES_StackPointer_t StackPtr, size_t StackSize, CFE_ES_TaskPriority_Atom_t Priority, uint32 Flags)`
Creates a new task under an existing Application.
- `CFE_Status_t CFE_ES_GetTaskIDByName (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName)`
Get a Task ID associated with a specified Task name.
- `CFE_Status_t CFE_ES_GetTaskName (char *TaskName, CFE_ES_TaskId_t TaskId, size_t BufferLength)`
Get a Task name for a specified Task ID.
- `CFE_Status_t CFE_ES_DeleteChildTask (CFE_ES_TaskId_t TaskId)`
Deletes a task under an existing Application.
- `void CFE_ES_ExitChildTask (void)`
Exits a child task.
- `void CFE_ES_BackgroundWakeUp (void)`
Wakes up the CFE background task.
- `CFE_Status_t CFE_ES_WriteToSysLog (const char *SpecStringPtr,...) OS_PRINTF(1`
Write a string to the cFE System Log.
- `CFE_Status_t uint32 CFE_ES_CalculateCRC (const void *DataPtr, size_t DataLength, uint32 InputCRC, CFE_ES_CrcType_Enum_t TypeCRC)`
Calculate a CRC on a block of memory.
- `void CFE_ES_ProcessAsyncEvent (void)`
Notification that an asynchronous event was detected by the underlying OS/PSP.
- `CFE_Status_t CFE_ES_RegisterCDS (CFE_ES_CDSHandle_t *CDSHandlePtr, size_t BlockSize, const char *Name)`
Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS).
- `CFE_Status_t CFE_ES_GetCDSBlockIDByName (CFE_ES_CDSHandle_t *BlockIdPtr, const char *BlockName)`
Get a CDS Block ID associated with a specified CDS Block name.
- `CFE_Status_t CFE_ES_GetCDSBlockName (char *BlockName, CFE_ES_CDSHandle_t BlockId, size_t BufferLength)`
Get a Block name for a specified Block ID.
- `CFE_Status_t CFE_ES_CopyToCDS (CFE_ES_CDSHandle_t Handle, const void *DataToCopy)`
Save a block of data in the Critical Data Store (CDS).
- `CFE_Status_t CFE_ES_RestoreFromCDS (void *RestoreToMemory, CFE_ES_CDSHandle_t Handle)`
Recover a block of data from the Critical Data Store (CDS).
- `CFE_Status_t CFE_ES_PoolCreateNoSem (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`
Initializes a memory pool created by an application without using a semaphore during processing.
- `CFE_Status_t CFE_ES_PoolCreate (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`
Initializes a memory pool created by an application while using a semaphore during processing.

- `CFE_Status_t CFE_ES_PoolCreateEx (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size, uint16 NumBlockSizes, const size_t *BlockSizes, bool UseMutex)`
Initializes a memory pool created by an application with application specified block sizes.
- `int32 CFE_ES_PoolDelete (CFE_ES_MemHandle_t PoolID)`
Deletes a memory pool that was previously created.
- `int32 CFE_ES_GetPoolBuf (CFE_ES_MemPoolBuf_t *BufPtr, CFE_ES_MemHandle_t Handle, size_t Size)`
Gets a buffer from the memory pool created by `CFE_ES_PoolCreate` or `CFE_ES_PoolCreateNoSem`.
- `CFE_Status_t CFE_ES_GetPoolBufInfo (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`
Gets info on a buffer previously allocated via `CFE_ES_GetPoolBuf`.
- `int32 CFE_ES_PutPoolBuf (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`
Releases a buffer from the memory pool that was previously allocated via `CFE_ES_GetPoolBuf`.
- `CFE_Status_t 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)`
Adds a new entry to the data buffer.
- `CFE_Status_t CFE_ES_RegisterGenCounter (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`
Register a generic counter.
- `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId)`
Delete a generic counter.
- `CFE_Status_t CFE_ES_IncrementGenCounter (CFE_ES_CounterId_t CounterId)`
Increments the specified generic counter.
- `CFE_Status_t CFE_ES_SetGenCount (CFE_ES_CounterId_t CounterId, uint32 Count)`
Set the specified generic counter.
- `CFE_Status_t CFE_ES_GetGenCount (CFE_ES_CounterId_t CounterId, uint32 *Count)`
Get the specified generic counter count.
- `CFE_Status_t CFE_ES_GetGenCounterIDByName (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`
Get the Id associated with a generic counter name.
- `CFE_Status_t CFE_ES_GetGenCounterName (char *CounterName, CFE_ES_CounterId_t CounterId, size_t BufferLength)`
Get a Counter name for a specified Counter ID.

12.83.1 Detailed Description

Purpose: Unit specification for Executive Services library functions and macros.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide
Notes:

12.83.2 Macro Definition Documentation

12.83.2.1 CFE_ES_DBIT #define CFE_ES_DBIT(
 x) (1L << (x)) /* Places a one at bit positions 0 thru 31 */
Definition at line 57 of file cfe_es.h.

12.83.2.2 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 58 of file cfe_es.h.

12.83.2.3 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 59 of file cfe_es.h.

12.83.2.4 OS_PRINTF #define OS_PRINTF(
m,
 n)

Definition at line 50 of file cfe_es.h.

12.84 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_es_extern_typedefs.h"
```

Data Structures

- union **CFE_ES_PoolAlign**

Pool Alignment.

Macros

- #define **CFE_ES_STATIC_POOL_TYPE**(size)
Static Pool Type.
- #define **CFE_ES_MEMPOOLBUF_C**(*x*) ((**CFE_ES_MemPoolBuf_t**)(*x*))
Conversion macro to create buffer pointer from another type.
- #define **CFE_ES_NO_MUTEX** false
Indicates that the memory pool selection will not use a semaphore.
- #define **CFE_ES_USE_MUTEX** true
Indicates that the memory pool selection will use a semaphore.

Reset Type extensions

- #define **CFE_ES_APP_RESTART** CFE_PSP_RST_TYPE_MAX

Conversions for ES resource IDs

- #define **CFE_ES_APPID_C**(*val*) ((**CFE_ES_AppId_t**)CFE_RESOURCEID_WRAP(*val*))
- #define **CFE_ES_TASKID_C**(*val*) ((**CFE_ES_TaskId_t**)CFE_RESOURCEID_WRAP(*val*))
- #define **CFE_ES_LIBID_C**(*val*) ((**CFE_ES_LibId_t**)CFE_RESOURCEID_WRAP(*val*))
- #define **CFE_ES_COUNTERID_C**(*val*) ((**CFE_ES_CounterId_t**)CFE_RESOURCEID_WRAP(*val*))
- #define **CFE_ES_MEMHANDLE_C**(*val*) ((**CFE_ES_MemHandle_t**)CFE_RESOURCEID_WRAP(*val*))
- #define **CFE_ES_CDHANDLE_C**(*val*) ((**CFE_ES_CDHandle_t**)CFE_RESOURCEID_WRAP(*val*))

Type-specific initializers for "undefined" resource IDs

- #define CFE_ES_APPID_UNDEFINED CFE_ES_APPID_C(CFE_RESOURCEID_UNDEFINED)
- #define CFE_ES_TASKID_UNDEFINED CFE_ES_TASKID_C(CFE_RESOURCEID_UNDEFINED)
- #define CFE_ES_LIBID_UNDEFINED CFE_ES_LIBID_C(CFE_RESOURCEID_UNDEFINED)
- #define CFE_ES_COUNTERID_UNDEFINED CFE_ES_COUNTERID_C(CFE_RESOURCEID_UNDEFINED)
- #define CFE_ES_MEMHANDLE_UNDEFINED CFE_ES_MEMHANDLE_C(CFE_RESOURCEID_UNDEFINED)
- #define CFE_ES_CDS_BAD_HANDLE CFE_ES_CDSHANDLE_C(CFE_RESOURCEID_UNDEFINED)

Task Stack Constants

- #define CFE_ES_TASK_STACK_ALLOCATE NULL /* aka OS_TASK_STACK_ALLOCATE in proposed O↔SAL change */

Indicates that the stack for the child task should be dynamically allocated.

Typedefs

- typedef void(* CFE_ES_TaskEntryFuncPtr_t) (void)
Required Prototype of Task Main Functions.
- typedef int32(* CFE_ES_LibraryEntryFuncPtr_t) (CFE_ES_LibId_t LibId)
Required Prototype of Library Initialization Functions.
- typedef CFE_ES_TaskEntryFuncPtr_t CFE_ES_ChildTaskMainFuncPtr_t
Compatible typedef for ES child task entry point.
- typedef void * CFE_ES_StackPointer_t
Type for the stack pointer of tasks.
- typedef enum CFE_ES_CrcType_Enum CFE_ES_CrcType_Enum_t
Checksum/CRC algorithm identifiers.
- typedef union CFE_ES_PoolAlign CFE_ES_PoolAlign_t
Pool Alignment.
- typedef void * CFE_ES_MemPoolBuf_t
Pointer type used for memory pool API.

Enumerations

- enum CFE_ES_CrcType_Enum {
 CFE_ES_CrcType_NONE = 0, CFE_ES_CrcType_16_ARC = 1, CFE_ES_CrcType_MAX = 2, CFE_ES_CrcType_CRC_16
 = CFE_ES_CrcType_16_ARC,
 CFE_ES_CrcType_CRC_8 = CFE_ES_CrcType_MAX + 1, CFE_ES_CrcType_CRC_32 = CFE_ES_CrcType_↔
 MAX + 2 }
Checksum/CRC algorithm identifiers.

12.84.1 Detailed Description

Purpose: Unit specification for Executive Services library functions and macros.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide
Notes:

12.84.2 Macro Definition Documentation

12.84.2.1 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 57 of file cfe_es_api_typeDefs.h.

12.84.2.2 CFE_ES_APPID_C #define CFE_ES_APPID_C(
 val) (([CFE_ES_AppId_t](#))CFE_RESOURCEID_WRAP(val))

Definition at line 208 of file cfe_es_api_typedefs.h.

12.84.2.3 CFE_ES_APPID_UNDEFINED #define CFE_ES_APPID_UNDEFINED [CFE_ES_APPID_C\(CFE_RESOURCEID_UNDEFINED\)](#)

Definition at line 220 of file cfe_es_api_typedefs.h.

12.84.2.4 CFE_ES_CDS_BAD_HANDLE #define CFE_ES_CDS_BAD_HANDLE [CFE_ES_CDSHANDLE_C\(CFE_RESOURCEID_UNDEFINED\)](#)

Definition at line 225 of file cfe_es_api_typedefs.h.

12.84.2.5 CFE_ES_CDSHANDLE_C #define CFE_ES_CDSHANDLE_C(
 val) (([CFE_ES_CDShandle_t](#))CFE_RESOURCEID_WRAP(val))

Definition at line 213 of file cfe_es_api_typedefs.h.

12.84.2.6 CFE_ES_COUNTERID_C #define CFE_ES_COUNTERID_C(
 val) (([CFE_ES_CounterId_t](#))CFE_RESOURCEID_WRAP(val))

Definition at line 211 of file cfe_es_api_typedefs.h.

12.84.2.7 CFE_ES_COUNTERID_UNDEFINED #define CFE_ES_COUNTERID_UNDEFINED [CFE_ES_COUNTERID_C\(CFE_RESOURCEID_UNDEFINED\)](#)

Definition at line 223 of file cfe_es_api_typedefs.h.

12.84.2.8 CFE_ES_LIBID_C #define CFE_ES_LIBID_C(
 val) (([CFE_ES_LibId_t](#))CFE_RESOURCEID_WRAP(val))

Definition at line 210 of file cfe_es_api_typedefs.h.

12.84.2.9 CFE_ES_LIBID_UNDEFINED #define CFE_ES_LIBID_UNDEFINED [CFE_ES_LIBID_C\(CFE_RESOURCEID_UNDEFINED\)](#)

Definition at line 222 of file cfe_es_api_typedefs.h.

12.84.2.10 CFE_ES_MEMHANDLE_C #define CFE_ES_MEMHANDLE_C(
 val) (([CFE_ES_MemHandle_t](#))CFE_RESOURCEID_WRAP(val))

Definition at line 212 of file cfe_es_api_typedefs.h.

12.84.2.11 CFE_ES_MEMHANDLE_UNDEFINED #define CFE_ES_MEMHANDLE_UNDEFINED [CFE_ES_MEMHANDLE_C\(CFE_RESOURCEID_UNDEFINED\)](#)

Definition at line 224 of file cfe_es_api_typedefs.h.

12.84.2.12 CFE_ES_MEMPOOLBUF_C #define CFE_ES_MEMPOOLBUF_C(
 x) (([CFE_ES_MemPoolBuf_t](#))(x))

Conversion macro to create buffer pointer from another type.

In cases where the actual buffer pointer is computed, this macro aids in converting the computed address (typically an OSAL "cpuaddr" type) into a buffer pointer.

Note

Any address calculation needs to take machine alignment requirements into account.

Definition at line 193 of file cfe_es_api_typedefs.h.

12.84.2.13 CFE_ES_NO_MUTEX #define CFE_ES_NO_MUTEX false

Indicates that the memory pool selection will not use a semaphore.

Definition at line 240 of file cfe_es_api_typedefs.h.

12.84.2.14 CFE_ES_STATIC_POOL_TYPE #define CFE_ES_STATIC_POOL_TYPE (size)**Value:**

```
union
{
    CFE_ES_PoolAlign_t Align;
    uint8              Data[size];
}
```

Static Pool Type.

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 160 of file cfe_es_api_typedefs.h.

12.84.2.15 CFE_ES_TASK_STACK_ALLOCATE #define CFE_ES_TASK_STACK_ALLOCATE NULL /* aka OS_TA↔SK_STACK_ALLOCATE in proposed OSAL change */

Indicates that the stack for the child task should be dynamically allocated.

This value may be supplied as the Stack Pointer argument to CFE_ES_ChildTaskCreate() to indicate that the stack should be dynamically allocated.

Definition at line 237 of file cfe_es_api_typedefs.h.

12.84.2.16 CFE_ES_TASKID_C #define CFE_ES_TASKID_C(val) ((CFE_ES_TaskId_t)CFE_RESOURCEID_WRAP(val))

Definition at line 209 of file cfe_es_api_typedefs.h.

12.84.2.17 CFE_ES_TASKID_UNDEFINED #define CFE_ES_TASKID_UNDEFINED CFE_ES_TASKID_C(CFE_RESOURCEID_UNDEFINED)

Definition at line 221 of file cfe_es_api_typedefs.h.

12.84.2.18 CFE_ES_USE_MUTEX #define CFE_ES_USE_MUTEX true

Indicates that the memory pool selection will use a semaphore.

Definition at line 241 of file cfe_es_api_typedefs.h.

12.84.3 Typedef Documentation

12.84.3.1 CFE_ES_ChildTaskMainFuncPtr_t typedef CFE_ES_TaskEntryFuncPtr_t CFE_ES_ChildTaskMainFuncPtr_t

Compatible typedef for ES child task entry point.

All ES task functions (main + child) use the same entry point type.

Definition at line 77 of file cfe_es_api_typedefs.h.

12.84.3.2 CFE_ES_CrcType_Enum_t `typedef enum CFE_ES_CrcType_Enum CFE_ES_CrcType_Enum_t`
Checksum/CRC algorithm identifiers.

Currently only CFE_ES_CrcType_16_ARC is supported.

All defined CRC algorithms should include a check value, which is the accepted result of computing the CRC across the fixed string "123456789"

12.84.3.3 CFE_ES_LibraryEntryFuncPtr_t `typedef int32 (* CFE_ES_LibraryEntryFuncPtr_t) (CFE_ES_LibId_t LibId)`

Required Prototype of Library Initialization Functions.

Definition at line 69 of file cfe_es_api_typedefs.h.

12.84.3.4 CFE_ES_MemPoolBuf_t `typedef void* CFE_ES_MemPoolBuf_t`

Pointer type used for memory pool API.

This is used in the Get/Put API calls to refer to a pool buffer.

This pointer is expected to be type cast to the real object type after getting a new buffer. Using void* allows this type conversion to occur easily.

Note

Older versions of CFE implemented the API using a uint32*, which required explicit type casting everywhere it was called. Although the API type is now void* to make usage easier, the pool buffers are aligned to machine requirements - typically 64 bits.

Definition at line 181 of file cfe_es_api_typedefs.h.

12.84.3.5 CFE_ES_PoolAlign_t `typedef union CFE_ES_PoolAlign CFE_ES_PoolAlign_t`

Pool Alignment.

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.

12.84.3.6 CFE_ES_StackPointer_t `typedef void* CFE_ES_StackPointer_t`

Type for the stack pointer of tasks.

This type is used in the CFE ES task API.

Definition at line 84 of file cfe_es_api_typedefs.h.

12.84.3.7 CFE_ES_TaskEntryFuncPtr_t `typedef void(* CFE_ES_TaskEntryFuncPtr_t) (void)`

Required Prototype of Task Main Functions.

Definition at line 68 of file cfe_es_api_typedefs.h.

12.84.4 Enumeration Type Documentation

12.84.4.1 CFE_ES_CrcType_Enum `enum CFE_ES_CrcType_Enum`

Checksum/CRC algorithm identifiers.

Currently only CFE_ES_CrcType_16_ARC is supported.

All defined CRC algorithms should include a check value, which is the accepted result of computing the CRC across the fixed string "123456789"

Enumerator

CFE_ES_CrcType_NONE	Reserved placeholder value. No computation is performed, always returns 0 as a result.
CFE_ES_CrcType_16_ARC	Implementation of CRC-16/ARC. Polynomial: 0x8005 Initialization: 0x0000 Reflect Input/Output: true Check value: 0xbb3d XorOut: 0x0000
CFE_ES_CrcType_MAX	Placeholder for end of normal enumeration list This should reflect the number of algorithms defined.
CFE_ES_CrcType_CRC_16	CRC-16 historically implied CRC-16/ARC CRC-8 historically defined but not implemented, value must not be 0
CFE_ES_CrcType_CRC_8	CRC-32 historically defined but not implemented, value must not be 0
CFE_ES_CrcType_CRC_32	

Definition at line 94 of file cfe_es_api_typedefs.h.

12.85 cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_evs_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

Macros

- #define CFE_EVS_Send(E, T, ...) CFE_EVS_SendEvent((E), CFE_EVS_EventType_##T, __VA_ARGS__)
- #define CFE_EVS_SendDbg(E, ...) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)
- #define CFE_EVS_SendInfo(E, ...) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)
- #define CFE_EVS_SendErr(E, ...) CFE_EVS_Send(E, ERROR, __VA_ARGS__)
- #define CFE_EVS_SendCrit(E, ...) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)

Functions

- CFE_Status_t CFE_EVS_Register (const void *Filters, uint16 NumEventFilters, uint16 FilterScheme)
Register an application for receiving event services.
- CFE_Status_t CFE_EVS_SendEvent (uint16 EventID, CFE_EVS_EventType_Enum_t EventType, const char *Spec,...) OS_PRINTF(3)
Generate a software event.
- CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (uint16 EventID, CFE_EVS_EventType_Enum_t EventType, CFE_ES_AppId_t AppID, const char *Spec,...) OS_PRINTF(4)
Generate a software event given the specified Application ID.
- CFE_Status_t CFE_Status_t CFE_Status_t CFE_EVS_SendTimedEvent (CFE_TIME_SysTime_t Time, uint16 EventID, CFE_EVS_EventType_Enum_t EventType, const char *Spec,...) OS_PRINTF(4)
Generate a software event with a specific time tag.

- **CFE_Status_t CFE_EVS_ResetFilter (uint16 EventID)**
Resets the calling application's event filter for a single event ID.
- **CFE_Status_t CFE_EVS_ResetAllFilters (void)**
Resets all of the calling application's event filters.

12.85.1 Detailed Description

Title: Event Services API Application Library Header File

Purpose: Unit specification for Event services library functions and macros.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.85.2 Macro Definition Documentation

12.85.2.1 CFE_EVS_Send #define CFE_EVS_Send(
 E,
 T,
 ...) CFE_EVS_SendEvent ((E), CFE_EVS_EventType_##T, __VA_ARGS__)

Definition at line 46 of file cfe_evs.h.

12.85.2.2 CFE_EVS_SendCrit #define CFE_EVS_SendCrit(
 E,
 ...) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)

Definition at line 50 of file cfe_evs.h.

12.85.2.3 CFE_EVS_SendDbg #define CFE_EVS_SendDbg(
 E,
 ...) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)

Definition at line 47 of file cfe_evs.h.

12.85.2.4 CFE_EVS_SendErr #define CFE_EVS_SendErr(
 E,
 ...) CFE_EVS_Send(E, ERROR, __VA_ARGS__)

Definition at line 49 of file cfe_evs.h.

12.85.2.5 CFE_EVS_SendInfo #define CFE_EVS_SendInfo(
 E,
 ...) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)

Definition at line 48 of file cfe_evs.h.

12.86 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_evs_extern_typedefs.h"
```

Data Structures

- struct [CFE_EVS_BinFilter](#)

Event message filter definition structure.

Macros

Common Event Filter Mask Values

Message is sent if (previous event count) & MASK == 0

- #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](#) 0xFFFFC
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.

Typedefs

- typedef struct [CFE_EVS_BinFilter](#) [CFE_EVS_BinFilter_t](#)

Event message filter definition structure.

12.86.1 Detailed Description

Title: Event Services API Application Library Header File

Purpose: Unit specification for Event services library functions and macros.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.86.2 Macro Definition Documentation

12.86.2.1 [CFE_EVS_EVERY_FOURTH_ONE](#) #define [CFE_EVS_EVERY_FOURTH_ONE](#) 0x0003

Sends every fourth event message. All others are filtered.

Definition at line 54 of file cfe_evs_api_typedefs.h.

12.86.2.2 CFE_EVS_EVERY_OTHER_ONE #define CFE_EVS_EVERY_OTHER_ONE 0x0001
Sends every other event.
Definition at line 52 of file cfe_evs_api_typedefs.h.

12.86.2.3 CFE_EVS_EVERY_OTHER_TWO #define CFE_EVS_EVERY_OTHER_TWO 0x0002
Sends two, filters one, sends two, filters one, etc.
Definition at line 53 of file cfe_evs_api_typedefs.h.

12.86.2.4 CFE_EVS_FIRST_16_STOP #define CFE_EVS_FIRST_16_STOP 0xFFFF0
Sends the first 16 events. All remaining messages are filtered.
Definition at line 49 of file cfe_evs_api_typedefs.h.

12.86.2.5 CFE_EVS_FIRST_32_STOP #define CFE_EVS_FIRST_32_STOP 0xFFE0
Sends the first 32 events. All remaining messages are filtered.
Definition at line 50 of file cfe_evs_api_typedefs.h.

12.86.2.6 CFE_EVS_FIRST_4_STOP #define CFE_EVS_FIRST_4_STOP 0xFFFFC
Sends the first 4 events. All remaining messages are filtered.
Definition at line 47 of file cfe_evs_api_typedefs.h.

12.86.2.7 CFE_EVS_FIRST_64_STOP #define CFE_EVS_FIRST_64_STOP 0xFFC0
Sends the first 64 events. All remaining messages are filtered.
Definition at line 51 of file cfe_evs_api_typedefs.h.

12.86.2.8 CFE_EVS_FIRST_8_STOP #define CFE_EVS_FIRST_8_STOP 0xFFF8
Sends the first 8 events. All remaining messages are filtered.
Definition at line 48 of file cfe_evs_api_typedefs.h.

12.86.2.9 CFE_EVS_FIRST_ONE_STOP #define CFE_EVS_FIRST_ONE_STOP 0xFFFFF
Sends the first event. All remaining messages are filtered.
Definition at line 45 of file cfe_evs_api_typedefs.h.

12.86.2.10 CFE_EVS_FIRST_TWO_STOP #define CFE_EVS_FIRST_TWO_STOP 0xFFFFE
Sends the first 2 events. All remaining messages are filtered.
Definition at line 46 of file cfe_evs_api_typedefs.h.

12.86.2.11 CFE_EVS_NO_FILTER #define CFE_EVS_NO_FILTER 0x0000
Stops any filtering. All messages are sent.
Definition at line 44 of file cfe_evs_api_typedefs.h.

12.86.3 Typedef Documentation

12.86.3.1 CFE_EVS_BinFilter_t `typedef struct CFE_EVS_BinFilter CFE_EVS_BinFilter_t`
Event message filter definition structure.

12.87 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference

```
#include "common_types.h"
#include "osconfig.h"
#include "cfe_platform_cfg.h"
#include "cfe_error.h"
#include "cfe_fs_api_typedefs.h"
#include "cfe_fs_extern_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

Functions

- **CFE_Status_t CFE_FS_ReadHeader (CFE_FS_Header_t *Hdr, osal_id_t 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.
- **CFE_Status_t CFE_FS_WriteHeader (osal_id_t FileDes, CFE_FS_Header_t *Hdr)**
Write the specified Standard cFE File Header to the specified file.
- **CFE_Status_t CFE_FS_SetTimestamp (osal_id_t FileDes, CFE_TIME_SysTime_t NewTimestamp)**
Modifies the Time Stamp field in the Standard cFE File Header for the specified file.
- **const char * CFE_FS_GetDefaultMountPoint (CFE_FS_FileCategory_t FileCategory)**
Get the default virtual mount point for a file category.
- **const char * CFE_FS_GetDefaultExtension (CFE_FS_FileCategory_t FileCategory)**
Get the default filename extension for a file category.
- **int32 CFE_FS_ParseInputFileNameEx (char *OutputBuffer, const char *InputBuffer, size_t OutputBufSize, size_t InputBufSize, const char *DefaultInput, const char *DefaultPath, const char *DefaultExtension)**
Parse a filename input from an input buffer into a local buffer.
- **int32 CFE_FS_ParseInputFileName (char *OutputBuffer, const char *InputName, size_t OutputBufSize, CFE_FS_FileCategory_t FileCategory)**
Parse a filename string from the user into a local buffer.
- **CFE_Status_t CFE_FS_ExtractFilenameFromPath (const char *OriginalPath, char *FileNameOnly)**
Extracts the filename from a unix style path and filename string.
- **int32 CFE_FS_BackgroundFileDumpRequest (CFE_FS_FileWriteMetaData_t *Meta)**
Register a background file dump request.
- **bool CFE_FS_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t *Meta)**
Query if a background file write request is currently pending.

12.87.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.88 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference

```
#include "common_types.h"
#include "osconfig.h"
#include "cfe_mission_cfg.h"
#include "cfe_fs_extern_typedefs.h"
```

Data Structures

- struct [CFE_FS_FileWriteMetaData](#)

External Metadata/State object associated with background file writes.

Typedefs

- typedef bool(* [CFE_FS_FileWriteGetData_t](#)) (void *Meta, [uint32](#) RecordNum, void **Buffer, size_t *BufSize)
- typedef void(* [CFE_FS_FileWriteOnEvent_t](#)) (void *Meta, [CFE_FS_FileWriteEvent_t](#) Event, [int32](#) Status, [uint32](#) RecordNum, size_t BlockSize, size_t Position)
- typedef struct [CFE_FS_FileWriteMetaData](#) [CFE_FS_FileWriteMetaData_t](#)

External Metadata/State object associated with background file writes.

Enumerations

- enum [CFE_FS_FileCategory_t](#) {
 [CFE_FS_FileCategory_UNKNOWN](#), [CFE_FS_FileCategory_DYNAMIC_MODULE](#), [CFE_FS_FileCategory_BINARY_DATA_DUMP](#),
[CFE_FS_FileCategory_TEXT_LOG](#),
[CFE_FS_FileCategory_SCRIPT](#), [CFE_FS_FileCategory_TEMP](#), [CFE_FS_FileCategory_MAX](#) }
- Generalized file types/categories known to FS.*
- enum [CFE_FS_FileWriteEvent_t](#) {
 [CFE_FS_FileWriteEvent_UNDEFINED](#), [CFE_FS_FileWriteEvent_COMPLETE](#), [CFE_FS_FileWriteEvent_CREATE_ERROR](#),
[CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR](#),
[CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR](#), [CFE_FS_FileWriteEvent_MAX](#) }

12.88.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.88.2 Typedef Documentation

12.88.2.1 CFE_FS_FileWriteGetData_t `typedef bool(* CFE_FS_FileWriteGetData_t) (void *Meta, uint32 RecordNum, void **Buffer, size_t *BufSize)`
 Data Getter routine provided by requester
 Outputs a data block. Should return true if the file is complete (last record/EOF), otherwise return false.

Parameters

<code>in, out</code>	<code>Meta</code>	Pointer to the metadata object
<code>in</code>	<code>RecordNum</code>	Incrementing record number counter
<code>out</code>	<code>Buffer</code>	Pointer to buffer data block, should be set by implementation
<code>out</code>	<code>BufSize</code>	Pointer to buffer data size, should be set by implementation

Returns

End of file status

Return values

<code>true</code>	if at last data record, and output file should be closed
<code>false</code>	if not at last record, more data records to write

Note

The implementation of this function must always set the "Buffer" and "BufSize" outputs. If no data is available, they may be set to NULL and 0, respectively.

Definition at line 98 of file cfe_fs_api_typedefs.h.

12.88.2.2 CFE_FS_FileWriteMetaData_t `typedef struct CFE_FS_FileWriteMetaData CFE_FS_FileWriteMetaData_t`
External Metadata/State object associated with background file writes.

Applications intending to schedule background file write jobs should instantiate this object in static/global data memory. This keeps track of the state of the file write request(s).

12.88.2.3 CFE_FS_FileWriteOnEvent_t `typedef void(* CFE_FS_FileWriteOnEvent_t) (void *Meta, CFE_FS_FileWriteEvent_t Event, int32 Status, uint32 RecordNum, size_t BlockSize, size_t Position)`
Event generator routine provided by requester
Invoked from certain points in the file write process. Implementation may invoke [CFE_EVS_SendEvent\(\)](#) appropriately to inform of progress.**Parameters**

in,out	<i>Meta</i>	Pointer to the metadata object
in	<i>Event</i>	Generalized type of event to report (not actual event ID)
in	<i>Status</i>	Generalized status code (may be from OSAL or CFE)
in	<i>RecordNum</i>	Record number counter at which event occurred
in	<i>BlockSize</i>	Size of record being processed when event occurred (if applicable)
in	<i>Position</i>	File position/size when event occurred

Definition at line 114 of file cfe_fs_api_typedefs.h.

12.88.3 Enumeration Type Documentation**12.88.3.1 CFE_FS_FileCategory_t** `enum CFE_FS_FileCategory_t`

Generalized file types/categories known to FS.

This defines different categories of files, where they may reside in different default locations of the virtualized file system. This is different from, and should not be confused with, the "SubType" field in the FS header. This value is only used at runtime for FS APIs and should not actually appear in any output file or message.

Enumerator

<code>CFE_FS_FileCategory_UNKNOWN</code>	Placeholder, unknown file category
<code>CFE_FS_FileCategory_DYNAMIC_MODULE</code>	Dynamically loadable apps/libraries (e.g. .so, .o, .dll, etc)
<code>CFE_FS_FileCategory_BINARY_DATA_DUMP</code>	Binary log file generated by various data dump commands
<code>CFE_FS_FileCategory_TEXT_LOG</code>	Text-based log file generated by various commands
<code>CFE_FS_FileCategory_SCRIPT</code>	Text-based Script files (e.g. ES startup script)
<code>CFE_FS_FileCategory_TEMP</code>	Temporary/Ephemeral files
<code>CFE_FS_FileCategory_MAX</code>	Placeholder, keep last

Definition at line 49 of file cfe_fs_api_typedefs.h.

12.88.3.2 CFE_FS_FileWriteEvent_t enum CFE_FS_FileWriteEvent_t

Enumerator

CFE_FS_FileWriteEvent_UNDEFINED	
CFE_FS_FileWriteEvent_COMPLETE	File is completed successfully
CFE_FS_FileWriteEvent_CREATE_ERROR	Unable to create/open file
CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR	Unable to write FS header
CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR	Unable to write data record
CFE_FS_FileWriteEvent_MAX	

Definition at line 69 of file cfe_fs_api_typedefs.h.

12.89 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_msg_hdr.h"
#include "cfe_msg_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

Functions

- **CFE_Status_t CFE_MSG_Init (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId, CFE_MSG_Size_t Size)**

Initialize a message.
- **CFE_Status_t CFE_MSG_GetSize (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t *Size)**

Gets the total size of a message.
- **CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)**

Sets the total size of a message.
- **CFE_Status_t CFE_MSG.GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)**

Gets the message type.
- **CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)**

Sets the message type.
- **CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)**

Gets the message header version.
- **CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)**

Sets the message header version.
- **CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)**

Gets the message secondary header boolean.
- **CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)**

Sets the message secondary header boolean.
- **CFE_Status_t CFE_MSG_GetApId (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t *ApId)**

Gets the message application ID.

- `CFE_Status_t CFE_MSG_SetApId (CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t ApId)`
Sets the message application ID.
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`
Gets the message segmentation flag.
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`
Sets the message segmentation flag.
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`
Gets the message sequence count.
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`
Sets the message sequence count.
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`
Gets the next sequence count value (rolls over if appropriate)
- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`
Gets the message EDS version.
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`
Sets the message EDS version.
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`
Gets the message endian.
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`
Sets the message endian.
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`
Gets the message playback flag.
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`
Sets the message playback flag.
- `CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)`
Gets the message subsystem.
- `CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)`
Sets the message subsystem.
- `CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)`
Gets the message system.
- `CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)`
Sets the message system.
- `CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t *MsgPtr)`
Calculates and sets the checksum of a message.
- `CFE_Status_t CFE_MSG_ValidateChecksum (const CFE_MSG_Message_t *MsgPtr, bool *isValid)`
Validates the checksum of a message.
- `CFE_Status_t CFE_MSG_SetFcnCode (CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode)`
Sets the function code field in a message.
- `CFE_Status_t CFE_MSG_GetFcnCode (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *FcnCode)`
Code)

- Gets the function code field from a message.
- CFE_Status_t CFE_MSG_GetMsgTime (const CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t *Time)
 - Gets the time field from a message.
- CFE_Status_t CFE_MSG_SetMsgTime (CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t NewTime)
 - Sets the time field in a message.
- CFE_Status_t CFE_MSG_GetMsgId (const CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t *MsgId)
 - Gets the message id from a message.
- CFE_Status_t CFE_MSG_SetMsgId (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId)
 - Sets the message id bits in a message.
- CFE_Status_t CFE_MSG_GetTypeFromMsgId (CFE_SB_MsgId_t MsgId, CFE_MSG_Type_t *Type)
 - Gets message type using message ID.
- CFE_Status_t CFE_MSG_OriginationAction (CFE_MSG_Message_t *MsgPtr, size_t BufferSize, bool *IsAcceptable)
 - Perform any necessary actions on a newly-created message, prior to sending.
- CFE_Status_t CFE_MSG_VerificationAction (const CFE_MSG_Message_t *MsgPtr, size_t BufferSize, bool *IsAcceptable)
 - Checks message integrity/acceptability.

12.89.1 Detailed Description

Message access APIs

12.90 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
```

Macros

- #define CFE_MSG_BAD_ARGUMENT CFE_SB_BAD_ARGUMENT
 - Error - bad argument.
- #define CFE_MSG_NOT_IMPLEMENTED CFE_SB_NOT_IMPLEMENTED
 - Error - not implemented.
- #define CFE_MSG_WRONG_MSG_TYPE CFE_SB_WRONG_MSG_TYPE
 - Error - wrong type.

Typedefs

- typedef size_t CFE_MSG_Size_t
 - Message size, note CCSDS maximum is UINT16_MAX+7.
- typedef uint32 CFE_MSG_Checksum_t
 - Message checksum (Oversized to avoid redefine)
- typedef uint16 CFE_MSG_FcnCode_t
 - Message function code.
- typedef uint16 CFE_MSG_HeaderVersion_t
 - Message header version.
- typedef uint16 CFE_MSG_Apld_t
 - Message application ID.
- typedef uint16 CFE_MSG_SequenceCount_t

- `typedef uint16 CFE_MSG_EDSVersion_t`
Message EDS version.
- `typedef uint16 CFE_MSG_Subsystem_t`
Message subsystem.
- `typedef uint16 CFE_MSG_System_t`
Message system.
- `typedef enum CFE_MSG_Type CFE_MSG_Type_t`
Message type.
- `typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t`
Segmentation flags.
- `typedef enum CFE_MSG_Endian CFE_MSG_Endian_t`
Endian flag.
- `typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t`
Playback flag.
- `typedef struct CFE_MSG_Message CFE_MSG_Message_t`
cFS generic base message
- `typedef struct CFE_MSG_CommandHeader CFE_MSG_CommandHeader_t`
cFS command header
- `typedef struct CFE_MSG_TelemetryHeader CFE_MSG_TelemetryHeader_t`
cFS telemetry header

Enumerations

- `enum CFE_MSG_Type { CFE_MSG_Type_Invalid, CFE_MSG_Type_Cmd, CFE_MSG_Type_Tlm }`
Message type.
- `enum CFE_MSG_SegmentationFlag { CFE_MSG_SegFlag_Invalid, CFE_MSG_SegFlag_Continue, CFE_MSG_SegFlag_First, CFE_MSG_SegFlag_Last, CFE_MSG_SegFlag_Unsegmented }`
Segmentation flags.
- `enum CFE_MSG_Endian { CFE_MSG_Endian_Invalid, CFE_MSG_Endian_Big, CFE_MSG_Endian_Little }`
Endian flag.
- `enum CFE_MSG_PlaybackFlag { CFE_MSG_PlayFlag_Invalid, CFE_MSG_PlayFlag_Original, CFE_MSG_PlayFlag_Playback }`
Playback flag.

12.90.1 Detailed Description

Typedefs for Message API

- Separate from API so these can be adjusted for custom implementations

12.90.2 Macro Definition Documentation

12.90.2.1 CFE_MSG_BAD_ARGUMENT `#define CFE_MSG_BAD_ARGUMENT CFE_SB_BAD_ARGUMENT`
Error - bad argument.

Definition at line 39 of file cfe_msg_api_typedefs.h.

12.90.2.2 CFE_MSG_NOT_IMPLEMENTED #define CFE_MSG_NOT_IMPLEMENTED CFE_SB_NOT_IMPLEMENTED
Error - not implemented.
Definition at line 40 of file cfe_msg_api_typedefs.h.

12.90.2.3 CFE_MSG_WRONG_MSG_TYPE #define CFE_MSG_WRONG_MSG_TYPE CFE_SB_WRONG_MSG_TYPE
Error - wrong type.
Definition at line 41 of file cfe_msg_api_typedefs.h.

12.90.3 Typedef Documentation

12.90.3.1 CFE_MSG_Apid_t typedef uint16 CFE_MSG_Apid_t
Message application ID.
Definition at line 50 of file cfe_msg_api_typedefs.h.

12.90.3.2 CFE_MSG_Checksum_t typedef uint32 CFE_MSG_Checksum_t
Message checksum (Oversized to avoid redefine)
Definition at line 47 of file cfe_msg_api_typedefs.h.

12.90.3.3 CFE_MSG_CommandHeader_t typedef struct CFE_MSG_CommandHeader CFE_MSG_CommandHeader_t
cFS command header
Definition at line 107 of file cfe_msg_api_typedefs.h.

12.90.3.4 CFE_MSG_EDSVersion_t typedef uint16 CFE_MSG_EDSVersion_t
Message EDS version.
Definition at line 52 of file cfe_msg_api_typedefs.h.

12.90.3.5 CFE_MSG_Endian_t typedef enum CFE_MSG_Endian CFE_MSG_Endian_t
Endian flag.

12.90.3.6 CFE_MSG_FcnCode_t typedef uint16 CFE_MSG_FcnCode_t
Message function code.
Definition at line 48 of file cfe_msg_api_typedefs.h.

12.90.3.7 CFE_MSG_HeaderVersion_t typedef uint16 CFE_MSG_HeaderVersion_t
Message header version.
Definition at line 49 of file cfe_msg_api_typedefs.h.

12.90.3.8 CFE_MSG_Message_t typedef struct CFE_MSG_Message CFE_MSG_Message_t
cFS generic base message
Definition at line 102 of file cfe_msg_api_typedefs.h.

12.90.3.9 CFE_MSG_PlaybackFlag_t `typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t`
Playback flag.

12.90.3.10 CFE_MSG_SegmentationFlag_t `typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t`
Segmentation flags.

12.90.3.11 CFE_MSG_SequenceCount_t `typedef uint16 CFE_MSG_SequenceCount_t`
Message sequence count.
Definition at line 51 of file `cfe_msg_api_typedefs.h`.

12.90.3.12 CFE_MSG_Size_t `typedef size_t CFE_MSG_Size_t`
Message size, note CCSDS maximum is `UINT16_MAX+7`.
Definition at line 46 of file `cfe_msg_api_typedefs.h`.

12.90.3.13 CFE_MSG_Subsystem_t `typedef uint16 CFE_MSG_Subsystem_t`
Message subsystem.
Definition at line 53 of file `cfe_msg_api_typedefs.h`.

12.90.3.14 CFE_MSG_System_t `typedef uint16 CFE_MSG_System_t`
Message system.
Definition at line 54 of file `cfe_msg_api_typedefs.h`.

12.90.3.15 CFE_MSG_TelemetryHeader_t `typedef struct CFE_MSG_TelemetryHeader CFE_MSG_TelemetryHeader_t`
cFS telemetry header
Definition at line 112 of file `cfe_msg_api_typedefs.h`.

12.90.3.16 CFE_MSG_Type_t `typedef enum CFE_MSG_Type CFE_MSG_Type_t`
Message type.

12.90.4 Enumeration Type Documentation

12.90.4.1 CFE_MSG_Endian `enum CFE_MSG_Endian`
Endian flag.

Enumerator

<code>CFE_MSG_Endian_Invalid</code>	Invalid endian setting.
<code>CFE_MSG_Endian_Big</code>	Big endian.
<code>CFE_MSG_Endian_Little</code>	Little endian.

Definition at line 75 of file `cfe_msg_api_typedefs.h`.

12.90.4.2 CFE_MSG_PlaybackFlag enum CFE_MSG_PlaybackFlag

Playback flag.

Enumerator

CFE_MSG_PlayFlag_Invalid	Invalid playback setting.
CFE_MSG_PlayFlag_Original	Original.
CFE_MSG_PlayFlag_Playback	Playback.

Definition at line 83 of file cfe_msg_api_typedefs.h.

12.90.4.3 CFE_MSG_SegmentationFlag enum CFE_MSG_SegmentationFlag

Segmentation flags.

Enumerator

CFE_MSG_SegFlag_Invalid	Invalid segmentation flag.
CFE_MSG_SegFlag_Continue	Continuation segment of User Data.
CFE_MSG_SegFlag_First	First segment of User Data.
CFE_MSG_SegFlag_Last	Last segment of User Data.
CFE_MSG_SegFlag_Unsegmented	Unsegmented data.

Definition at line 65 of file cfe_msg_api_typedefs.h.

12.90.4.4 CFE_MSG_Type enum CFE_MSG_Type

Message type.

Enumerator

CFE_MSG_Type_Invalid	Message type invalid, undefined, not implemented.
CFE_MSG_Type_Cmd	Command message type.
CFE_MSG_Type_Tlm	Telemetry message type.

Definition at line 57 of file cfe_msg_api_typedefs.h.

12.91 cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference

```
#include "cfe_resourceid_api_typedefs.h"
```

Functions

- `uint32 CFE_ResourceId_GetBase (CFE_ResourceId_t ResourceId)`

Get the Base value (type/category) from a resource ID value.
- `uint32 CFE_ResourceId_GetSerial (CFE_ResourceId_t ResourceId)`

Get the Serial Number (sequential ID) from a resource ID value.
- `CFE_ResourceId_t CFE_ResourceId_FindNext (CFE_ResourceId_t StartId, uint32 TableSize, bool(*Check← Func)(CFE_ResourceId_t))`

Locate the next resource ID which does not map to an in-use table entry.

- `int32 CFE_Resourceld_ToIndex (CFE_Resourceld_t Id, uint32 BaseValue, uint32 TableSize, uint32 *Idx)`

Internal routine to aid in converting an ES resource ID to an array index.

Resource ID test/conversion macros and inline functions

- `#define CFE_RESOURCEID_TO ULONG(id) CFE_Resourceld_ToInteger(CFE_RESOURCEID_UNWRAP(id))`
Convert a derived (app-specific) ID directly into an "unsigned long".
- `#define CFE_RESOURCEID_TEST_DEFINED(id) CFE_Resourceld_IsDefined(CFE_RESOURCEID_UNWRAP(id))`
Determine if a derived (app-specific) ID is defined or not.
- `#define CFE_RESOURCEID_TEST_EQUAL(id1, id2) CFE_Resourceld_Equal(CFE_RESOURCEID_UNWRAP(id1), CFE_RESOURCEID_UNWRAP(id2))`
Determine if two derived (app-specific) IDs are equal.
- `static unsigned long CFE_Resourceld_ToInteger (CFE_Resourceld_t id)`
Convert a resource ID to an integer.
- `static CFE_Resourceld_t CFE_Resourceld_FromInteger (unsigned long Value)`
Convert an integer to a resource ID.
- `static bool CFE_Resourceld_Equal (CFE_Resourceld_t id1, CFE_Resourceld_t id2)`
Compare two Resource ID values for equality.
- `static bool CFE_Resourceld_IsDefined (CFE_Resourceld_t id)`
Check if a resource ID value is defined.

12.91.1 Detailed Description

Contains global prototypes and definitions related to resource management and related CFE resource IDs.
A CFE ES Resource ID is a common way to identify CFE-managed resources such as apps, tasks, counters, memory pools, CDS blocks, and other entities.

Simple operations are provided as inline functions, which should alleviate the need to do direct manipulation of resource IDs:

- Check for undefined ID value
- Check for equality of two ID values
- Convert ID to simple integer (typically for printing/logging)
- Convert simple integer to ID (inverse of above)

12.91.2 Macro Definition Documentation

12.91.2.1 CFE_RESOURCEID_TEST_DEFINED `#define CFE_RESOURCEID_TEST_DEFINED (id) CFE_Resourceld_IsDefined(CFE_RESOURCEID_UNWRAP(id))`

Determine if a derived (app-specific) ID is defined or not.

This generic routine is implemented as a macro so it is agnostic to the actual argument type, and it will evaluate correctly so long as the argument type is based on the CFE_RESOURCEID_BASE_TYPE.

Definition at line 70 of file cfe_resourceid.h.

```
12.91.2.2 CFE_RESOURCEID_TEST_EQUAL #define CFE_RESOURCEID_TEST_EQUAL( id1,
    id2 ) CFE_ResourceId_Equal(CFE_RESOURCEID_UNWRAP(id1), CFE_RESOURCEID_UNWRAP(id2))
```

Determine if two derived (app-specific) IDs are equal.

This generic routine is implemented as a macro so it is agnostic to the actual argument type, and it will evaluate correctly so long as the argument type is based on the CFE_RESOURCEID_BASE_TYPE.

Definition at line 78 of file cfe_resourceid.h.

```
12.91.2.3 CFE_RESOURCEID_TO ULONG #define CFE_RESOURCEID_TO ULONG( id ) CFE_ResourceId_ToInteger(CFE_RESOURCEID_UNWRAP(id))
```

Convert a derived (app-specific) ID directly into an "unsigned long".

This generic routine is implemented as a macro so it is agnostic to the actual argument type, and it will evaluate correctly so long as the argument type is based on the CFE_RESOURCEID_BASE_TYPE.

There is no inverse of this macro, as it depends on the actual derived type desired. Applications needing to recreate an ID from an integer should use [CFE_Resourceld_FromInteger\(\)](#) combined with a cast/conversion to the correct/intended derived type, as needed.

Note

This evaluates as an "unsigned long" such that it can be used in printf()-style functions with the "%lx" modifier without extra casting, as this is the most typical use-case for representing an ID as an integer.

Definition at line 62 of file cfe_resourceid.h.

12.91.3 Function Documentation

```
12.91.3.1 CFE_Resourceld_Equal() static bool CFE_ResourceId_Equal( CFE_ResourceId_t id1,
    CFE_ResourceId_t id2 ) [inline], [static]
```

Compare two Resource ID values for equality.

Parameters

in	<i>id1</i>	Resource ID to check
in	<i>id2</i>	Resource ID to check

Returns

true if id1 and id2 are equal, false otherwise.

Definition at line 133 of file cfe_resourceid.h.

Referenced by [CFE_Resourceld_IsDefined\(\)](#).

```
12.91.3.2 CFE_Resourceld_FindNext() CFE_ResourceId_t CFE_ResourceId_FindNext( CFE_ResourceId_t StartId,
    uint32 TableSize,
    bool(*)(CFE_ResourceId_t) CheckFunc )
```

Locate the next resource ID which does not map to an in-use table entry.

This begins searching from StartId which should be the most recently issued ID for the resource category. This will then search for the next ID which does *not* map to a table entry that is in use. That is, it does not alias any valid ID when converted to an array index.

returns an undefined ID value if no open slots are available

Parameters

in	<i>StartId</i>	the last issued ID for the resource category (app, lib, etc).
in	<i>TableSize</i>	the maximum size of the target table
in	<i>CheckFunc</i>	a function to check if the given ID is available

Returns

Next ID value which does not map to a valid entry

Return values

CFE_RESOURCEID_UNDEFINED	if no open slots or bad arguments.
--	------------------------------------

12.91.3.3 CFE_ResourceId_FromInteger() `static CFE_ResourceId_t CFE_ResourceId_FromInteger (`

`unsigned long Value) [inline], [static]`

Convert an integer to a resource ID.

This is the inverse of [CFE_ResourceId_ToInteger\(\)](#), and reconstitutes the original CFE_ResourceId_t value from the integer representation.

This may be used, for instance, where an ID value is parsed from a text file or message using C library APIs such as scanf() or strtoul().

See also

[CFE_ResourceId_ToInteger\(\)](#)

Parameters

in	<i>Value</i>	Integer value to convert
----	--------------	--------------------------

Returns

ID value corresponding to integer

Definition at line 121 of file cfe_resourceid.h.

12.91.3.4 CFE_ResourceId_GetBase() `uint32 CFE_ResourceId_GetBase (`

`CFE_ResourceId_t ResourceId)`

Get the Base value (type/category) from a resource ID value.

This masks out the ID serial number to obtain the base value, which is different for each resource type.

Note

The value is NOT shifted or otherwise adjusted.

Parameters

in	<i>Resource← Id</i>	the resource ID to decode
----	-------------------------	---------------------------

Returns

The base value associated with that ID

12.91.3.5 CFE_ResourcId_GetSerial() `uint32 CFE_ResourceId_GetSerial (`

`CFE_ResourceId_t ResourceId)`

Get the Serial Number (sequential ID) from a resource ID value.

This masks out the ID base value to obtain the serial number, which is different for each entity created.

Parameters

in	<i>ResourceId</i>	the resource ID to decode
	<i>Id</i>	

Returns

The serial number associated with that ID

12.91.3.6 CFE_ResourcId_IsDefined() `static bool CFE_ResourceId_IsDefined (`

`CFE_ResourceId_t id) [inline], [static]`

Check if a resource ID value is defined.

The constant `CFE_RESOURCEID_UNDEFINED` represents an undefined ID value, such that the expression:

`CFE_ResourceId_IsDefined(CFE_RESOURCEID_UNDEFINED)`

Always returns false.

Parameters

in	<i>id</i>	Resource ID to check

Returns

True if the ID may refer to a defined entity, false if invalid/undefined.

Definition at line 151 of file `cfe_resourceid.h`.

References `CFE_ResourcId_Equal()`, and `CFE_RESOURCEID_UNDEFINED`.

Here is the call graph for this function:

**12.91.3.7 CFE_ResourcId_ToIndex()** `int32 CFE_ResourceId_ToIndex (`

`CFE_ResourceId_t Id,`

```

    uint32 BaseValue,
    uint32 TableSize,
    uint32 * Idx )

```

Internal routine to aid in converting an ES resource ID to an array index.

Parameters

in	<i>Id</i>	The resource ID
in	<i>BaseValue</i>	The respective ID base value corresponding to the ID type
in	<i>TableSize</i>	The actual size of the internal table (MAX index value + 1)
out	<i>Idx</i>	The output index

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

12.91.3.8 CFE_ResourceId_ToInteger()

```
static unsigned long CFE_ResourceId_ToInteger (
```

```
    CFE_ResourceId_t id ) [inline], [static]
```

Convert a resource ID to an integer.

This is primarily intended for logging purposes, such as writing to debug console, event messages, or log files, using printf-like APIs.

For compatibility with C library APIs, this returns an "unsigned long" type and should be used with the "%lx" format specifier in a printf format string.

Note

No assumptions should be made about the actual integer value, such as its base/range. It may be printed, but should not be modified or tested/compared using other arithmetic ops, and should never be used as the index to an array or table. See the related function [CFE_ResourceId_ToIndex\(\)](#) for cases where a zero-based array/table index is needed.

See also

[CFE_ResourceId_FromInteger\(\)](#)

Parameters

in	<i>id</i>	Resource ID to convert
----	-----------	------------------------

Returns

Integer value corresponding to ID

Definition at line 102 of file cfe_resourceid.h.

12.92 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference

```
#include "cfe_resourceid_typedef.h"
```

Macros

Resource ID predefined values

- #define **CFE_RESOURCEID_UNDEFINED** ((CFE_ResourceId_t)CFE_RESOURCEID_WRAP(0))
A resource ID value that represents an undefined/unused resource.
- #define **CFE_RESOURCEID_RESERVED** ((CFE_ResourceId_t)CFE_RESOURCEID_WRAP(0xFFFFFFFF))
A resource ID value that represents a reserved entry.

12.92.1 Detailed Description

Contains global prototypes and definitions related to resource management and related CFE resource IDs.
A CFE ES Resource ID is a common way to identify CFE-managed resources such as apps, tasks, counters, memory pools, CDS blocks, and other entities.

Simple operations are provided as inline functions, which should alleviate the need to do direct manipulation of resource IDs:

- Check for undefined ID value
- Check for equality of two ID values
- Convert ID to simple integer (typically for printing/logging)
- Convert simple integer to ID (inverse of above)

12.92.2 Macro Definition Documentation

12.92.2.1 CFE_RESOURCEID_RESERVED #define CFE_RESOURCEID_RESERVED ((CFE_ResourceId_t)CFE_RESOURCEID_WRAP(0xFFFFFFFF))

A resource ID value that represents a reserved entry.

This is not a valid value for any resource type, but is used to mark table entries that are not available for use. For instance, this may be used while setting up an entry initially.

Definition at line 74 of file cfe_resourceid_api_typedefs.h.

12.92.2.2 CFE_RESOURCEID_UNDEFINED #define CFE_RESOURCEID_UNDEFINED ((CFE_ResourceId_t)CFE_RESOURCEID_WRAP(0))

A resource ID value that represents an undefined/unused resource.

This constant may be used to initialize local variables of the CFE_ResourceId_t type to a safe value that will not alias a valid ID.

By design, this value is also the result of zeroing a CFE_ResourceId_t type via standard functions like memset(), such that objects initialized using this method will also be set to safe values.

Definition at line 65 of file cfe_resourceid_api_typedefs.h.

12.93 cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_sb_api_typedefs.h"
```

```
#include "cfe_es_api_typedefs.h"
```

Macros

- `#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

Functions

- `CFE_Status_t CFE_SB_CreatePipe (CFE_SB_Pipeld_t *PipeldPtr, uint16 Depth, const char *PipeName)`
Creates a new software bus pipe.
- `CFE_Status_t CFE_SB_DeletePipe (CFE_SB_Pipeld_t Pipeld)`
Delete a software bus pipe.
- `CFE_Status_t CFE_SB_Pipeld_ToIndex (CFE_SB_Pipeld_t PipeID, uint32 *Idx)`
Obtain an index value correlating to an SB Pipe ID.
- `CFE_Status_t CFE_SB_SetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 Opts)`
Set options on a pipe.
- `CFE_Status_t CFE_SB_GetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 *OptsPtr)`
Get options on a pipe.
- `CFE_Status_t CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_Pipeld_t Pipeld)`
Get the pipe name for a given id.
- `CFE_Status_t CFE_SB_GetPipeldByName (CFE_SB_Pipeld_t *PipeldPtr, const char *PipeName)`
Get pipe id by pipe name.
- `CFE_Status_t CFE_SB_SubscribeEx (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, CFE_SB_Qos_t Quality, uint16 MsgLim)`
Subscribe to a message on the software bus.
- `CFE_Status_t CFE_SB_Subscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`
Subscribe to a message on the software bus with default parameters.
- `CFE_Status_t CFE_SB_SubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, uint16 MsgLim)`
Subscribe to a message while keeping the request local to a cpu.
- `CFE_Status_t CFE_SB_Unsubscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`
Remove a subscription to a message on the software bus.
- `CFE_Status_t CFE_SB_UnsubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`
Remove a subscription to a message on the software bus on the current CPU.
- `CFE_Status_t CFE_SB_TransmitMsg (const CFE_MSG_Message_t *MsgPtr, bool IsOrigination)`
Transmit a message.
- `CFE_Status_t CFE_SB_ReceiveBuffer (CFE_SB_Buffer_t **BufPtr, CFE_SB_Pipeld_t Pipeld, int32 TimeOut)`
Receive a message from a software bus pipe.
- `CFE_SB_Buffer_t * CFE_SB_AllocateMessageBuffer (size_t MsgSize)`
Get a buffer pointer to use for "zero copy" SB sends.
- `CFE_Status_t CFE_SB_ReleaseMessageBuffer (CFE_SB_Buffer_t *BufPtr)`

Release an unused "zero copy" buffer pointer.

- `CFE_Status_t CFE_SB_TransmitBuffer (CFE_SB_Buffer_t *BufPtr, bool IsOrigination)`
Transmit a buffer.
- `void CFE_SB_SetUserDataLength (CFE_MSG_Message_t *MsgPtr, size_t DataLength)`
Sets the length of user data in a software bus message.
- `void CFE_SB_TimeStampMsg (CFE_MSG_Message_t *MsgPtr)`
Sets the time field in a software bus message with the current spacecraft time.
- `int32 CFE_SB_MessageStringSet (char *DestStringPtr, const char *SourceStringPtr, size_t DestMaxSize, size_t SourceMaxSize)`
Copies a string into a software bus message.
- `void * CFE_SB_GetUserData (CFE_MSG_Message_t *MsgPtr)`
Get a pointer to the user data portion of a software bus message.
- `size_t CFE_SB_GetUserDataLength (const CFE_MSG_Message_t *MsgPtr)`
Gets the length of user data in a software bus message.
- `int32 CFE_SB_MessageStringGet (char *DestStringPtr, const char *SourceStringPtr, const char *DefaultString, size_t DestMaxSize, size_t SourceMaxSize)`
Copies a string out of a software bus message.
- `bool CFE_SB_IsValidMsgId (CFE_SB_MsgId_t MsgId)`
Identifies whether a given `CFE_SB_MsgId_t` is valid.
- `static bool CFE_SB_MsgId_Equal (CFE_SB_MsgId_t MsgId1, CFE_SB_MsgId_t MsgId2)`
Identifies whether 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`.
- `CFE_SB_MsgId_Atom_t CFE_SB_CmdTopicIdToMsgId (uint16 TopicId, uint16 InstanceNum)`
Converts a topic ID and instance number combination into a MsgID value integer.
- `CFE_SB_MsgId_Atom_t CFE_SB_TlmTopicIdToMsgId (uint16 TopicId, uint16 InstanceNum)`
Converts a topic ID and instance number combination into a MsgID value integer.
- `CFE_SB_MsgId_Atom_t CFE_SB_GlobalCmdTopicIdToMsgId (uint16 TopicId)`
Converts a topic ID to a MsgID value integer for Global commands.
- `CFE_SB_MsgId_Atom_t CFE_SB_GlobalTlmTopicIdToMsgId (uint16 TopicId)`
Converts a topic ID to a MsgID value integer for Global telemetry.
- `CFE_SB_MsgId_Atom_t CFE_SB_LocalCmdTopicIdToMsgId (uint16 TopicId)`
Converts a topic ID to a MsgID value integer for local commands.
- `CFE_SB_MsgId_Atom_t CFE_SB_LocalTlmTopicIdToMsgId (uint16 TopicId)`
Converts a topic ID to a MsgID value integer for local telemetry.

12.93.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.
Author: R.McGraw/SSI

12.93.2 Macro Definition Documentation

12.93.2.1 CFE_BIT `#define CFE_BIT(`
 `x) (1 << (x))`

Places a one at bit positions 0 - 31.

Definition at line 44 of file cfe_sb.h.

12.93.2.2 CFE_CLR `#define CFE_CLR(`
 `i,`
 `x) ((i) &= ~CFE_BIT(x))`

Clears bit x of i.

Definition at line 46 of file cfe_sb.h.

12.93.2.3 CFE_SET `#define CFE_SET(`
 `i,`
 `x) ((i) |= CFE_BIT(x))`

Sets bit x of i.

Definition at line 45 of file cfe_sb.h.

12.93.2.4 CFE_TST `#define CFE_TST(`
 `i,`
 `x) (((i)&CFE_BIT(x)) != 0)`

true(non zero) if bit x of i is set

Definition at line 47 of file cfe_sb.h.

12.94 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_sb_extern_typedefs.h"
#include "cfe_msg_api_typedefs.h"
#include "cfe_resourceid_api_typedefs.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- union **CFE_SB_Msg**
Software Bus generic message.

Macros

- `#define CFE_SB_POLL 0`
Option used with `CFE_SB_ReceiveBuffer` to request immediate pipe status.
- `#define CFE_SB_PEND_FOREVER -1`
Option used with `CFE_SB_ReceiveBuffer` to force a wait for next message.
- `#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_MSGID_WRAP_VALUE(val)`
Translation macro to convert from MsgId integer values to opaque/abstract API values.

- `#define CFE_SB_MSGID_C(val) ((CFE_SB_MsgId_t)CFE_SB_MSGID_WRAP_VALUE(val))`
Translation macro to convert to MsgId integer values from a literal.
- `#define CFE_SB_MSGID_UNWRAP_VALUE(mid) ((mid).Value)`
Translation macro to convert to MsgId integer values from opaque/abstract API values.
- `#define CFE_SB_MSGID_RESERVED CFE_SB_MSGID_WRAP_VALUE(0)`
Reserved value for `CFE_SB_MsgId_t` that will not match any valid MsgId.
- `#define CFE_SB_INVALID_MSG_ID CFE_SB_MSGID_C(0)`
A literal of the `CFE_SB_MsgId_t` type representing an invalid ID.
- `#define CFE_SB_PIPEID_C(val) ((CFE_SB_Pipeld_t)CFE_RESOURCEID_WRAP(val))`
Cast/Convert a generic `CFE_ResourceId_t` to a `CFE_SB_Pipeld_t`.
- `#define CFE_SB_INVALID_PIPE CFE_SB_PIPEID_C(CFE_RESOURCEID_UNDEFINED)`
A `CFE_SB_Pipeld_t` value which is always invalid.
- `#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_DEFAULT_QOS ((CFE_SB_Qos_t) {0})`
Default Qos macro.

Typedefs

- `typedef union CFE_SB_Msg CFE_SB_Buffer_t`
Software Bus generic message.

12.94.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.
Author: R.McGraw/SSI

12.94.2 Macro Definition Documentation

12.94.2.1 CFE_SB_DEFAULT_QOS `#define CFE_SB_DEFAULT_QOS ((CFE_SB_Qos_t) {0})`

Default Qos macro.

Definition at line 135 of file `cfe_sb_api_typedefs.h`.

12.94.2.2 CFE_SB_INVALID_MSG_ID `#define CFE_SB_INVALID_MSG_ID CFE_SB_MSGID_C(0)`

A literal of the `CFE_SB_MsgId_t` type representing an invalid ID.

This value should be used for runtime initialization of `CFE_SB_MsgId_t` values.

Note

This may be a compound literal in a future revision. Per C99, compound literals are lvalues, not rvalues, so this value should not be used in static/compile-time data initialization. For static data initialization purposes (rvalue), `CFE_SB_MSGID_RESERVED` should be used instead. However, in the current implementation, they are equivalent.

Definition at line 113 of file `cfe_sb_api_typedefs.h`.

12.94.2.3 CFE_SB_INVALID_PIPE #define CFE_SB_INVALID_PIPE CFE_SB_PIPEID_C(CFE_RESOURCEID_UNDEFINED)
A CFE_SB_PipeId_t value which is always invalid.

This may be used as a safe initializer for CFE_SB_PipeId_t values

Definition at line 125 of file cfe_sb_api_typedefs.h.

12.94.2.4 CFE_SB_MSGID_C #define CFE_SB_MSGID_C(
 val) ((CFE_SB_MsgId_t)CFE_SB_MSGID_WRAP_VALUE(val))

Translation macro to convert to MsgId integer values from a literal.

This ensures that the literal is interpreted as the CFE_SB_MsgId_t type, rather than the default type associated with that literal (e.g. int/unsigned int).

Note

Due to constraints in C99 this style of initializer can only be used at runtime, not for static/compile-time initializers.

See also

[CFE_SB_ValueToMsgId\(\)](#)

Definition at line 80 of file cfe_sb_api_typedefs.h.

12.94.2.5 CFE_SB_MSGID_RESERVED #define CFE_SB_MSGID_RESERVED CFE_SB_MSGID_WRAP_VALUE(0)

Reserved value for CFE_SB_MsgId_t that will not match any valid MsgId.

This rvalue macro can be used for static/compile-time data initialization to ensure that the initialized value does not alias to a valid MsgId object.

Definition at line 100 of file cfe_sb_api_typedefs.h.

12.94.2.6 CFE_SB_MSGID_UNWRAP_VALUE #define CFE_SB_MSGID_UNWRAP_VALUE(
 mid) ((mid).Value)

Translation macro to convert to MsgId integer values from opaque/abstract API values.

This conversion exists in macro form to allow compile-time evaluation for constants, and should not be used directly in application code.

For applications, use the [CFE_SB_MsgIdToValue\(\)](#) inline function instead.

See also

[CFE_SB_MsgIdToValue\(\)](#)

Definition at line 92 of file cfe_sb_api_typedefs.h.

12.94.2.7 CFE_SB_MSGID_WRAP_VALUE #define CFE_SB_MSGID_WRAP_VALUE(
 val)

Value:

```
{           \
    val      \
}
```

Translation macro to convert from MsgId integer values to opaque/abstract API values.

This conversion exists in macro form to allow compile-time evaluation for constants, and should not be used directly in application code.

For applications, use the [CFE_SB_ValueToMsgId\(\)](#) inline function instead.

See also

[CFE_SB_ValueToMsgId\(\)](#)

Definition at line 64 of file cfe_sb_api_typedefs.h.

12.94.2.8 CFE_SB_PEND_FOREVER #define CFE_SB_PEND_FOREVER -1
Option used with [CFE_SB_ReceiveBuffer](#) to force a wait for next message.
Definition at line 46 of file cfe_sb_api_typedefs.h.

12.94.2.9 CFE_SB_PIPEID_C #define CFE_SB_PIPEID_C(
 val) (([CFE_SB_PipeId_t](#))CFE_RESOURCEID_WRAP(val))
Cast/Convert a generic CFE_ResourceId_t to a CFE_SB_PipeId_t.
Definition at line 118 of file cfe_sb_api_typedefs.h.

12.94.2.10 CFE_SB_POLL #define CFE_SB_POLL 0
Option used with [CFE_SB_ReceiveBuffer](#) to request immediate pipe status.
Definition at line 45 of file cfe_sb_api_typedefs.h.

12.94.2.11 CFE_SB_SUBSCRIPTION #define CFE_SB_SUBSCRIPTION 0
Subtype specifier used in [CFE_SB_SingleSubscriptionTlm_t](#) by SBN App.
Definition at line 47 of file cfe_sb_api_typedefs.h.

12.94.2.12 CFE_SB_UNSUBSCRIPTION #define CFE_SB_UNSUBSCRIPTION 1
Subtype specified used in [CFE_SB_SingleSubscriptionTlm_t](#) by SBN App.
Definition at line 48 of file cfe_sb_api_typedefs.h.

12.94.3 Typedef Documentation

12.94.3.1 CFE_SB_Buffer_t typedef union [CFE_SB_Msg](#) CFE_SB_Buffer_t
Software Bus generic message.

12.95 cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_tbl_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
```

Functions

- [CFE_Status_t CFE_TBL_Register](#) ([CFE_TBL_Handle_t](#) *TblHandlePtr, const char *Name, size_t Size, uint16 TblOptionFlags, [CFE_TBL_CallbackFuncPtr_t](#) TblValidationFuncPtr)
Register a table with cFE to obtain Table Management Services.
- [CFE_Status_t CFE_TBL_Share](#) ([CFE_TBL_Handle_t](#) *TblHandlePtr, const char *TblName)
Obtain handle of table registered by another application.
- [CFE_Status_t CFE_TBL_Unregister](#) ([CFE_TBL_Handle_t](#) TblHandle)
Unregister a table.
- [CFE_Status_t 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.

- [CFE_Status_t CFE_TBL_Update \(CFE_TBL_Handle_t TblHandle\)](#)
Update contents of a specified table, if an update is pending.
- [CFE_Status_t CFE_TBL_Validate \(CFE_TBL_Handle_t TblHandle\)](#)
Perform steps to validate the contents of a table image.
- [CFE_Status_t CFE_TBL_Manage \(CFE_TBL_Handle_t TblHandle\)](#)
Perform standard operations to maintain a table.
- [CFE_Status_t CFE_TBL_DumpToBuffer \(CFE_TBL_Handle_t TblHandle\)](#)
Copies the contents of a Dump Only Table to a shared buffer.
- [CFE_Status_t CFE_TBL_Modified \(CFE_TBL_Handle_t TblHandle\)](#)
Notify cFE Table Services that table contents have been modified by the Application.
- [CFE_Status_t CFE_TBL_GetAddress \(void **TblPtr, CFE_TBL_Handle_t TblHandle\)](#)
Obtain the current address of the contents of the specified table.
- [CFE_Status_t CFE_TBL_ReleaseAddress \(CFE_TBL_Handle_t TblHandle\)](#)
Release previously obtained pointer to the contents of the specified table.
- [CFE_Status_t CFE_TBL_GetAddresses \(void **TblPtrs\[\], uint16 NumTables, const CFE_TBL_Handle_t TblHandles\[\]\)](#)
Obtain the current addresses of an array of specified tables.
- [CFE_Status_t CFE_TBL_ReleaseAddresses \(uint16 NumTables, const CFE_TBL_Handle_t TblHandles\[\]\)](#)
Release the addresses of an array of specified tables.
- [CFE_Status_t CFE_TBL_GetStatus \(CFE_TBL_Handle_t TblHandle\)](#)
Obtain current status of pending actions for a table.
- [CFE_Status_t CFE_TBL_GetInfo \(CFE_TBL_Info_t *TblInfoPtr, const char *TblName\)](#)
Obtain characteristics/information of/about a specified table.
- [CFE_Status_t CFE_TBL_NotifyByMessage \(CFE_TBL_Handle_t TblHandle, CFE_SB_MsgId_t MsgId, CFE_MSG_FcnCode_t CommandCode, uint32 Parameter\)](#)
Instruct cFE Table Services to notify Application via message when table requires management.

12.95.1 Detailed Description

Title: Table Services API Application Library Header File

Purpose: Unit specification for Table services library functions and macros.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

Notes:

12.96 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_time_extern_typedefs.h"
```

Data Structures

- struct [CFE_TBL_Info](#)

Table Info.

Macros

- `#define CFE_TBL_OPT_BUFFER_MSK (0x0001)`
Table buffer mask.
- `#define CFE_TBL_OPT_SNGL_BUFFER (0x0000)`
Single buffer table.
- `#define CFE_TBL_OPT_DBL_BUFFER (0x0001)`
Double buffer table.
- `#define CFE_TBL_OPT_LD_DMP_MSK (0x0002)`
Table load/dump mask.
- `#define CFE_TBL_OPT_LOAD_DUMP (0x0000)`
Load/Dump table.
- `#define CFE_TBL_OPT_DUMP_ONLY (0x0002)`
Dump only table.
- `#define CFE_TBL_OPT_USR_DEF_MSK (0x0004)`
Table user defined mask.
- `#define CFE_TBL_OPT_NOT_USR_DEF (0x0000)`
Not user defined table.
- `#define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)`
User Defined table.,
- `#define CFE_TBL_OPT_CRITICAL_MSK (0x0008)`
Table critical mask.
- `#define CFE_TBL_OPT_NOT_CRITICAL (0x0000)`
Not critical table.
- `#define CFE_TBL_OPT_CRITICAL (0x0008)`
Critical table.
- `#define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)`
Default table options.
- `#define CFE_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_FULL_NAME_LEN)`
Table maximum full name length.
- `#define CFE_TBL_BAD_TABLE_HANDLE (CFE_TBL_Handle_t)0xFFFF`
Bad table handle.

Typedefs

- `typedef int32(* CFE_TBL_CallbackFuncPtr_t) (void *TblPtr)`
Table Callback Function.
- `typedef int16 CFE_TBL_Handle_t`
Table Handle primitive.
- `typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t`
Table Source.
- `typedef struct CFE_TBL_Info CFE_TBL_Info_t`
Table Info.

Enumerations

- `enum CFE_TBL_SrcEnum { CFE_TBL_SRC_FILE = 0, CFE_TBL_SRC_ADDRESS }`
Table Source.

12.96.1 Detailed Description

Title: Table Services API Application Library Header File

Purpose: Unit specification for Table services library functions and macros.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

Notes:

12.96.2 Macro Definition Documentation

12.96.2.1 CFE_TBL_BAD_TABLE_HANDLE #define CFE_TBL_BAD_TABLE_HANDLE (CFE_TBL_Handle_t) 0xFFFF

Bad table handle.

Definition at line 79 of file cfe_tbl_api_typedefs.h.

12.96.2.2 CFE_TBL_MAX_FULL_NAME_LEN #define CFE_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_FULL_NAME_LEN)

Table maximum full name length.

The full length of table names is defined at the mission scope. This is defined here to support applications that depend on [cfe_tbl.h](#) providing this value.

Definition at line 76 of file cfe_tbl_api_typedefs.h.

12.96.3 Typedef Documentation

12.96.3.1 CFE_TBL_CallbackFuncPtr_t typedef int32(* CFE_TBL_CallbackFuncPtr_t) (void *TblPtr)

Table Callback Function.

Definition at line 84 of file cfe_tbl_api_typedefs.h.

12.96.3.2 CFE_TBL_Handle_t typedef int16 CFE_TBL_Handle_t

Table Handle primitive.

Definition at line 87 of file cfe_tbl_api_typedefs.h.

12.96.3.3 CFE_TBL_Info_t typedef struct CFE_TBL_Info CFE_TBL_Info_t

Table Info.

12.96.3.4 CFE_TBL_SrcEnum_t typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t

Table Source.

12.96.4 Enumeration Type Documentation

12.96.4.1 CFE_TBL_SrcEnum enum CFE_TBL_SrcEnum

Table Source.

Enumerator

CFE_TBL_SRC_FILE	File source When this option is selected, the <code>SrcDataPtr</code> 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	Address source When this option is selected, the <code>SrcDataPtr</code> 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 90 of file `cfe_tbl_api_typedefs.h`.

12.97 cfe/modules/core_api/fsw/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](#)

Table File summary object.

Macros

- #define [CFE_TBL_FILEDEF](#)(ObjName, TblName, Desc, Filename) `CFE_TBL_FileDef_t CFE_TBL_FileDef = {#ObjName "\0", #TblName "\0", #Desc "\0", #Filename "\0", sizeof(ObjName)}`;

Macro to assist in with table definition object declaration.

Typedefs

- typedef struct [CFE_TBL_FileDef](#) `CFE_TBL_FileDef_t`

Table File summary object.

12.97.1 Detailed Description

Title: ELF2CFETBL Utility Header File for Table Images

Purpose: This header file provides a data structure definition and macro definition required in source code that is intended to be compiled into a cFE compatible Table Image file.

Design Notes:

Typically, a user would include this file in a ".c" file that contains nothing but a desired instantiation of values for a table image along with the macro defined below. After compilation, the resultant elf file can be processed using the 'elf2cfetbl' utility to generate a file that can be loaded onto a cFE flight system and successfully loaded into a table using the cFE Table Services.

References: Flight Software Branch C Coding Standard Version 1.0a

Notes:

12.97.2 Macro Definition Documentation

```
12.97.2.1 CFE_TBL_FILEDEF #define CFE_TBL_FILEDEF (
    ObjName,
    TblName,
    Desc,
    Filename ) CFE_TBL_FileDef_t CFE_TBL_FileDef = { #ObjName "\0", #TblName "\0", #Desc
"\0", #Filename "\0", sizeof(ObjName) };
```

Macro to assist in with table definition object declaration.

See notes in the [CFE_TBL_FileDef_t](#) structure type about naming conventions and recommended practices for the various fields.

The CFE_TBL_FILEDEF macro can be used to simplify the declaration of a table image when using the elf2cfetbl utility. Note that the macro adds a NULL at the end to ensure that it is null-terminated. (C allows a struct to be statically initialized with a string exactly the length of the array, which loses the null terminator.) This means the actual length limit of the fields are the above LEN - 1.

An example of the source code and how this macro would be used is as follows:

```
#include "cfe_tbl_filedef.h"
typedef struct MyTblStruct
{
    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 149 of file [cfe_tbl_filedef.h](#).

12.97.3 Typedef Documentation

12.97.3.1 CFE_TBL_FileDef_t [typedef struct CFE_TBL_FileDef CFE_TBL_FileDef_t](#)

Table File summary object.

The definition of the file definition metadata that can be used by external tools (e.g. elf2cfetbl) to generate CFE table data files.

12.98 cfe/modules/core_api/fsw/inc/cfe_time.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_time_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
```

Macros

- [#define CFE_TIME_Copy\(m, t\)](#)

Time Copy.

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 (MET + SCTF) time.

- [CFE_TIME_SysTime_t CFE_TIME_GetUTC \(void\)](#)

Get the current UTC (MET + SCTF - Leap Seconds) time.

- `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.
- `CFE_TIME_SysTime_t CFE_TIME_GetSTCF` (void)
Get the current value of the spacecraft time correction factor (STCF).
- `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.
- `CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (CFE_TIME_SysTime_t METTime)`
Convert specified MET into Spacecraft Time.
- `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.
- `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.
- `CFE_Status_t CFE_TIME_RegisterSynchCallback (CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)`
Registers a callback function that is called whenever time synchronization occurs.
- `CFE_Status_t CFE_TIME_UnregisterSynchCallback (CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)`
Unregisters a callback function that is called whenever time synchronization occurs.
- `void CFE_TIME_Print (char *PrintBuffer, CFE_TIME_SysTime_t TimeToPrint)`
Print a time value as a string.
- `void CFE_TIME_Local1HzISR` (void)
This function is called via a timer callback set up at initialization of the TIME service.

12.98.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.98.2 Macro Definition Documentation

12.98.2.1 CFE_TIME_Copy #define CFE_TIME_Copy (
 m,
 t)

Value:

```
{
    (m) ->Seconds      = (t) ->Seconds;      \
    (m) ->Subseconds  = (t) ->Subseconds;  \
}
```

Time Copy.

Macro to copy systime into another systime. Preferred to use this macro as it does not require the two arguments to be exactly the same type, it will work with any two structures that define "Seconds" and "Subseconds" members.
 Definition at line 48 of file cfe_time.h.

12.99 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_time_extern_typedefs.h"
```

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_ZERO_VALUE ((CFE_TIME_SysTime_t){0, 0})

Typedefs

- typedef enum CFE_TIME_Compare CFE_TIME_Compare_t
Enumerated types identifying the relative relationships of two times.
- typedef int32(* CFE_TIME_SynchCallbackPtr_t) (void)
Time Synchronization Callback Function Ptr Type.

Enumerations

- enum CFE_TIME_Compare { 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.

12.99.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.99.2 Macro Definition Documentation

12.99.2.1 CFE_TIME_PRINTED_STRING_SIZE #define CFE_TIME_PRINTED_STRING_SIZE 24

Required size of buffer to be passed into CFE_TIME_Print (includes null terminator)

Definition at line 44 of file cfe_time_api_typedefs.h.

12.99.2.2 CFE_TIME_ZERO_VALUE #define CFE_TIME_ZERO_VALUE ((CFE_TIME_SysTime_t){0, 0})

A general-purpose initializer for CFE_TIME_SysTime_t values.

Represents "time zero" in the CFE_TIME_SysTime_t domain. This can be used as a general purpose initializer for instantiations of the CFE_TIME_SysTime_t type.

Definition at line 54 of file cfe_time_api_typedefs.h.

12.99.3 Typedef Documentation

12.99.3.1 CFE_TIME_Compare_t typedef enum CFE_TIME_Compare CFE_TIME_Compare_t

Enumerated types identifying the relative relationships of two times.

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.

12.99.3.2 CFE_TIME_SynchCallbackPtr_t typedef int32 (* CFE_TIME_SynchCallbackPtr_t) (void)

Time Synchronization Callback Function Ptr Type.

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 84 of file cfe_time_api_typedefs.h.

12.99.4 Enumeration Type Documentation

12.99.4.1 CFE_TIME_Compare enum CFE_TIME_Compare

Enumerated types identifying the relative relationships of two times.

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 69 of file cfe_time_api_typedefs.h.

12.100 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference

Macros

- `#define CFE_BUILD_NUMBER 235`
Development: Number of development git commits since CFE_BUILD_BASELINE.
- `#define CFE_BUILD_BASELINE "equuleus-rc1"`
Development: Reference git tag for build number.
- `#define CFE_BUILD_DEV_CYCLE "equuleus-rc2"`
Development: Release name for current development cycle.
- `#define CFE_BUILD_CODENAME "Equuleus"`
: Development: Code name for the current build
- `#define CFE_MAJOR_VERSION 6`
Major version number.
- `#define CFE_MINOR_VERSION 7`
Minor version number.
- `#define CFE_REVISION 0`
Revision version number. Value of 0 indicates a development version.
- `#define CFE_LAST_OFFICIAL "v6.7.0"`
Last official release.
- `#define CFE_MISSION_REV 0xFF`
Mission revision.
- `#define CFE_STR_HELPER(x) #x`
Convert argument to string.
- `#define CFE_STR(x) CFE_STR_HELPER(x)`
Expand macro before conversion.
- `#define CFE_SRC_VERSION CFE_BUILD_BASELINE "+dev" CFE_STR(CFE_BUILD_NUMBER)`
Short Build Version String.
- `#define CFE_CFG_MAX_VERSION_STR_LEN 256`
Max Version String length.

12.100.1 Detailed Description

Provide version identifiers for the cFE core. See [Version Numbers](#) for further details.

12.100.2 Macro Definition Documentation

12.100.2.1 CFE_BUILD_BASELINE `#define CFE_BUILD_BASELINE "equuleus-rc1"`

Development: Reference git tag for build number.

Definition at line 30 of file cfe_version.h.

12.100.2.2 CFE_BUILD_CODENAME `#define CFE_BUILD_CODENAME "Equuleus"`

: Development: Code name for the current build

Definition at line 32 of file cfe_version.h.

12.100.2.3 CFE_BUILD_DEV_CYCLE #define CFE_BUILD_DEV_CYCLE "equuleus-rc2"

Development: Release name for current development cycle.

Definition at line 31 of file cfe_version.h.

12.100.2.4 CFE_BUILD_NUMBER #define CFE_BUILD_NUMBER 235

Development: Number of development git commits since CFE_BUILD_BASELINE.

Definition at line 29 of file cfe_version.h.

12.100.2.5 CFE_CFG_MAX_VERSION_STR_LEN #define CFE_CFG_MAX_VERSION_STR_LEN 256

Max Version String length.

Maximum length that a cFE version string can be.

Definition at line 69 of file cfe_version.h.

12.100.2.6 CFE_LAST_OFFICIAL #define CFE_LAST_OFFICIAL "v6.7.0"

Last official release.

Definition at line 42 of file cfe_version.h.

12.100.2.7 CFE_MAJOR_VERSION #define CFE_MAJOR_VERSION 6

Major version number.

Definition at line 35 of file cfe_version.h.

12.100.2.8 CFE_MINOR_VERSION #define CFE_MINOR_VERSION 7

Minor version number.

Definition at line 36 of file cfe_version.h.

12.100.2.9 CFE_MISSION_REV #define CFE_MISSION_REV 0xFF

Mission revision.

Values 1-254 are reserved for mission use to denote patches/customizations as needed. NOTE: Reserving 0 and 0xFF for cFS open-source development use (pending resolution of nasa/cFS#440)

Definition at line 51 of file cfe_version.h.

12.100.2.10 CFE_REVISION #define CFE_REVISION 0

Revision version number. Value of 0 indicates a development version.

Definition at line 37 of file cfe_version.h.

12.100.2.11 CFE_SRC_VERSION #define CFE_SRC_VERSION CFE_BUILD_BASELINE "+dev" CFE_STR(CFE_BUILD_NUMBER)

Short Build Version String.

Short string identifying the build, see [Version Numbers](#) for suggested format for development and official releases.

Definition at line 62 of file cfe_version.h.

12.100.2.12 CFE_STR #define CFE_STR(
 x) CFE_STR_HELPER(x)

Expand macro before conversion.

Definition at line 54 of file cfe_version.h.

12.100.2.13 CFE_STR_HELPER #define CFE_STR_HELPER(
 x) #x

Convert argument to string.

Definition at line 53 of file cfe_version.h.

12.101 cfe/modules/es/config/default_cfe_es_extern_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_typedef.h"
#include "cfe_mission_cfg.h"
```

Data Structures

- struct [CFE_ES_AppInfo](#)
Application Information.
- struct [CFE_ES_TaskInfo](#)
Task Information.
- struct [CFE_ES_CDSRegDumpRec](#)
CDS Register Dump Record.
- struct [CFE_ES_BlockStats](#)
Block statistics.
- struct [CFE_ES_MemPoolStats](#)
Memory Pool Statistics.

Macros

- #define [CFE_ES_MEMOFFSET_C](#)(x) (([CFE_ES_MemOffset_t](#))(x))
Memory Offset initializer wrapper.
- #define [CFE_ES_MEMOFFSET_TO_SIZE_T](#)(x) (([size_t](#))(x))
Memory Offset to integer value ([size_t](#)) wrapper.
- #define [CFE_ES_MEMADDRESS_C](#)(x) (([CFE_ES_MemAddress_t](#))(([cpuaddr](#))(x)&0xFFFFFFFF))
Memory Address initializer wrapper.
- #define [CFE_ES_MEMADDRESS_TO_PTR](#)(x) (([void *](#))([cpuaddr](#))(x))
Memory Address to pointer wrapper.

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.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_AppId_t**
A type for Application IDs.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_TaskId_t**
A type for Task IDs.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_LibId_t**
A type for Library IDs.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CounterId_t**
A type for Counter IDs.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_MemHandle_t**
Memory Handle type.
- **typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CDSHandle_t**
CDS Handle type.
- **typedef uint16 CFE_ES_TaskPriority_Atom_t**
Type used for task priority in CFE ES as including the commands/telemetry messages.
- **typedef uint32 CFE_ES_MemOffset_t**
Type used for memory sizes and offsets in commands and telemetry.
- **typedef uint32 CFE_ES_MemAddress_t**
Type used for memory addresses in command and telemetry messages.
- **typedef struct CFE_ES_AppInfo CFE_ES_AppInfo_t**
Application Information.
- **typedef struct CFE_ES_TaskInfo CFE_ES_TaskInfo_t**
Task Information.
- **typedef struct CFE_ES_CDSRegDumpRec CFE_ES_CDSRegDumpRec_t**
CDS Register Dump Record.
- **typedef struct CFE_ES_BlockStats CFE_ES_BlockStats_t**
Block statistics.
- **typedef struct CFE_ES_MemPoolStats CFE_ES_MemPoolStats_t**
Memory Pool Statistics.

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, CFE_ES_AppType_LIBRARY = 3 }**
Label definitions associated with CFE_ES_AppType_Enum_t.

- enum `CFE_ES_RunStatus` {
 `CFE_ES_RunStatus_UNDEFINED` = 0, `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, `CFE_ES_RunStatus_MAX` }
- 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, `CFE_ES_SystemState_MAX` }
- 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.*

12.101.1 Detailed Description

Declarations and prototypes for `cfe_es_extern_typedefs` module

12.101.2 Macro Definition Documentation

12.101.2.1 `CFE_ES_MEMADDRESS_C` `#define CFE_ES_MEMADDRESS_C(` `x) ((CFE_ES_MemAddress_t)((cpuaddr)(x)&0xFFFFFFFF))`

Memory Address initializer wrapper.

A converter macro to use when initializing a `CFE_ES_MemAddress_t` from a pointer value of a different type.

Definition at line 417 of file `default_cfe_es_extern_typedefs.h`.

12.101.2.2 `CFE_ES_MEMADDRESS_TO_PTR` `#define CFE_ES_MEMADDRESS_TO_PTR(` `x) ((void *)(cpuaddr)(x))`

Memory Address to pointer wrapper.

A converter macro to use when interpreting a `CFE_ES_MemAddress_t` as a pointer value.

Definition at line 425 of file `default_cfe_es_extern_typedefs.h`.

12.101.2.3 `CFE_ES_MEMOFFSET_C` `#define CFE_ES_MEMOFFSET_C(` `x) ((CFE_ES_MemOffset_t)(x))`

Memory Offset initializer wrapper.

A converter macro to use when initializing a `CFE_ES_MemOffset_t` from an integer value of a different type.

Definition at line 380 of file `default_cfe_es_extern_typedefs.h`.

12.101.2.4 CFE_ES_MEMOFFSET_TO_SIZE_T `#define CFE_ES_MEMOFFSET_TO_SIZE_T(x) ((size_t)(x))`

Memory Offset to integer value (size_t) wrapper.

A converter macro to use when interpreting a CFE_ES_MemOffset_t value as a "size_t" type
Definition at line 388 of file default_cfe_es_extern_typedefs.h.

12.101.3 Typedef Documentation

12.101.3.1 CFE_ES_AppId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_AppId_t`

A type for Application IDs.

This is the type that is used for any API accepting or returning an App ID

Definition at line 312 of file default_cfe_es_extern_typedefs.h.

12.101.3.2 CFE_ES_AppInfo_t `typedef struct CFE_ES_AppInfo CFE_ES_AppInfo_t`

Application Information.

Structure that is used to provide information about an app. It is primarily used for the QueryOne and QueryAll Commands.

While this structure is primarily intended for Application info, it can also represent Library information where only a subset of the information applies.

12.101.3.3 CFE_ES_AppState_Enum_t `typedef uint32 CFE_ES_AppState_Enum_t`

Application Run State.

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 305 of file default_cfe_es_extern_typedefs.h.

12.101.3.4 CFE_ES_AppType_Enum_t `typedef uint8 CFE_ES_AppType_Enum_t`

Identifies type of CFE application.

See also

enum [CFE_ES_AppType](#)

Definition at line 104 of file default_cfe_es_extern_typedefs.h.

12.101.3.5 CFE_ES_BlockStats_t `typedef struct CFE_ES_BlockStats CFE_ES_BlockStats_t`

Block statistics.

Sub-Structure that is used to provide information about a specific block size/bucket within a memory pool.

12.101.3.6 CFE_ES_CDSHandle_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CDSHandle_t`

CDS Handle type.

Data type used to hold Handles of Critical Data Stores. See [CFE_ES_RegisterCDS](#)

Definition at line 348 of file default_cfe_es_extern_typedefs.h.

12.101.3.7 CFE_ES_CDSRegDumpRec_t `typedef struct CFE_ES_CDSRegDumpRec CFE_ES_CDSRegDumpRec_t`
CDS Register Dump Record.

Structure that is used to provide information about a critical data store. It is primarily used for the Dump CDS registry ([CFE_ES_DUMP_CDS_REGISTRY_CC](#)) command.

Note

There is not currently a telemetry message directly containing this data structure, but it does define the format of the data file generated by the Dump CDS registry command. Therefore it should be considered part of the overall telemetry interface.

12.101.3.8 CFE_ES_CounterId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CounterId_t`

A type for Counter IDs.

This is the type that is used for any API accepting or returning a Counter ID

Definition at line 333 of file default_cfe_es_extern_typedefs.h.

12.101.3.9 CFE_ES_ExceptionAction_Enum_t `typedef uint8 CFE_ES_ExceptionAction_Enum_t`

Identifies action to take if exception occurs.

See also

enum [CFE_ES_ExceptionAction](#)

Definition at line 76 of file default_cfe_es_extern_typedefs.h.

12.101.3.10 CFE_ES_LibId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_LibId_t`

A type for Library IDs.

This is the type that is used for any API accepting or returning a Lib ID

Definition at line 326 of file default_cfe_es_extern_typedefs.h.

12.101.3.11 CFE_ES_LogEntryType_Enum_t `typedef uint8 CFE_ES_LogEntryType_Enum_t`

Type of entry in the Error and Reset (ER) Log.

See also

enum [CFE_ES_LogEntryType](#)

Definition at line 252 of file default_cfe_es_extern_typedefs.h.

12.101.3.12 CFE_ES_LogMode_Enum_t `typedef uint8 CFE_ES_LogMode_Enum_t`

Identifies handling of log messages after storage is filled.

See also

enum [CFE_ES_LogMode](#)

Definition at line 53 of file default_cfe_es_extern_typedefs.h.

12.101.3.13 CFE_ES_MemAddress_t `typedef uint32 CFE_ES_MemAddress_t`

Type used for memory addresses in command and telemetry messages.

For backward compatibility with existing CFE code this should be uint32, but if running on a 64-bit platform, addresses in telemetry will be truncated to 32 bits and therefore will not be valid.

On 64-bit platforms this can be a 64-bit address which will allow the full memory address in commands and telemetry, but this will break compatibility with existing control systems, and may also change the alignment/padding of messages. In either case this must be an unsigned type.

FSW code should access this value via the macros provided, which converts to the native "cpuaddr" type provided by OSAL. This macro provides independence between the message representation and local representation of a memory address.

Definition at line 409 of file default_cfe_es_extern_typedefs.h.

12.101.3.14 CFE_ES_MemHandle_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_MemHandle_t`

Memory Handle type.

Data type used to hold Handles of Memory Pools created via CFE_ES_PoolCreate and CFE_ES_PoolCreateNoSem

Definition at line 341 of file default_cfe_es_extern_typedefs.h.

12.101.3.15 CFE_ES_MemOffset_t `typedef uint32 CFE_ES_MemOffset_t`

Type used for memory sizes and offsets in commands and telemetry.

For backward compatibility with existing CFE code this should be uint32, but all telemetry information will be limited to 4GB in size as a result.

On 64-bit platforms this can be a 64-bit value which will allow larger memory objects, but this will break compatibility with existing control systems, and may also change the alignment/padding of messages.

In either case this must be an unsigned type.

Definition at line 372 of file default_cfe_es_extern_typedefs.h.

12.101.3.16 CFE_ES_MemPoolStats_t `typedef struct CFE_ES_MemPoolStats CFE_ES_MemPoolStats_t`

Memory Pool Statistics.

Structure that is used to provide information about a memory pool. Used by the Memory Pool Stats telemetry message.

See also

[CFE_ES_SEND_MEM_POOL_STATS_CC](#)

12.101.3.17 CFE_ES_RunStatus_Enum_t `typedef uint32 CFE_ES_RunStatus_Enum_t`

Run Status and Exit Status identifiers.

See also

enum [CFE_ES_RunStatus](#)

Definition at line 172 of file default_cfe_es_extern_typedefs.h.

12.101.3.18 CFE_ES_SystemState_Enum_t `typedef uint32 CFE_ES_SystemState_Enum_t`

The overall cFE System State.

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 229 of file default_cfe_es_extern_typedefs.h.

12.101.3.19 CFE_ES_TaskId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_TaskId_t`

A type for Task IDs.

This is the type that is used for any API accepting or returning a Task ID

Definition at line 319 of file default_cfe_es_extern_typedefs.h.

12.101.3.20 CFE_ES_TaskInfo_t `typedef struct CFE_ES_TaskInfo CFE_ES_TaskInfo_t`

Task Information.

Structure that is used to provide information about a task. It is primarily used for the Query All Tasks ([CFE_ES_QUERY_ALL_TASKS_CC](#)) command.

Note

There is not currently a telemetry message directly containing this data structure, but it does define the format of the data file generated by the Query All Tasks command. Therefore it should be considered part of the overall telemetry interface.

12.101.3.21 CFE_ES_TaskPriority_Atom_t `typedef uint16 CFE_ES_TaskPriority_Atom_t`

Type used for task priority in CFE ES as including the commands/telemetry messages.

Note

the valid range is only 0-255 (same as OSAL) but a wider type is used for backward compatibility in binary formats of messages.

Definition at line 358 of file default_cfe_es_extern_typedefs.h.

12.101.4 Enumeration Type Documentation**12.101.4.1 CFE_ES_AppState** `enum CFE_ES_AppState`

Label definitions associated with `CFE_ES_AppState_Enum_t`.

Enumerator

<code>CFE_ES_AppState_UNDEFINED</code>	Initial state before app thread is started.
<code>CFE_ES_AppState_EARLY_INIT</code>	App thread has started, app performing early initialization of its own data.
<code>CFE_ES_AppState_LATE_INIT</code>	Early/Local initialization is complete. First sync point.
<code>CFE_ES_AppState_RUNNING</code>	All initialization is complete. Second sync point.
<code>CFE_ES_AppState_WAITING</code>	Application is waiting on a Restart/Reload/Delete request.
<code>CFE_ES_AppState_STOPPED</code>	Application is stopped.
<code>CFE_ES_AppState_MAX</code>	Reserved entry, marker for the maximum state.

Definition at line 257 of file default_cfe_es_extern_typedefs.h.

12.101.4.2 CFE_ES_AppType enum [CFE_ES_AppType](#)

Label definitions associated with CFE_ES_AppType_Enum_t.

Enumerator

CFE_ES_AppType_CORE	CFE core application.
CFE_ES_AppType_EXTERNAL	CFE external application.
CFE_ES_AppType_LIBRARY	CFE library.

Definition at line 81 of file default_cfe_es_extern_typedefs.h.

12.101.4.3 CFE_ES_ExceptionAction enum [CFE_ES_ExceptionAction](#)

Label definitions associated with CFE_ES_ExceptionAction_Enum_t.

Enumerator

CFE_ES_ExceptionAction_RESTART_APP	Restart application if exception occurs.
CFE_ES_ExceptionAction_PROC_RESTART	Restart processor if exception occurs.

Definition at line 58 of file default_cfe_es_extern_typedefs.h.

12.101.4.4 CFE_ES_LogEntryType enum [CFE_ES_LogEntryType](#)

Label definitions associated with CFE_ES_LogEntryType_Enum_t.

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 default_cfe_es_extern_typedefs.h.

12.101.4.5 CFE_ES_LogMode enum [CFE_ES_LogMode](#)

Label definitions associated with CFE_ES_LogMode_Enum_t.

Enumerator

CFE_ES_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_ES_LogMode_DISCARD	Discard Log Mode.

Definition at line 35 of file default_cfe_es_extern_typedefs.h.

12.101.4.6 CFE_ES_RunStatus enum [CFE_ES_RunStatus](#)

Label definitions associated with CFE_ES_RunStatus_Enum_t.

Enumerator

CFE_ES_RunStatus_UNDEFINED	Reserved value, should not be used.
----------------------------	-------------------------------------

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.
CFE_ES_RunStatus_MAX	Reserved value, marker for the maximum state.

Definition at line 109 of file default_cfe_es_extern_typedefs.h.

12.101.4.7 CFE_ES_SystemState enum CFE_ES_SystemState

Label definitions associated with CFE_ES_SystemState_Enum_t.

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
CFE_ES_SystemState_MAX	Reserved value, marker for the maximum state.

Definition at line 177 of file default_cfe_es_extern_typedefs.h.

12.102 cfe/modules/es/config/default_cfe_es_fcncodes.h File Reference

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_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_SYS_LOG_CC 10
- #define CFE_ES_WRITE_SYS_LOG_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_SYS_LOG_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

12.102.1 Detailed Description

Specification for the CFE Executive Services (CFE_ES) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.102.2 Macro Definition Documentation

12.102.2.1 CFE_ES_CLEAR_ER_LOG_CC #define CFE_ES_CLEAR_ER_LOG_CC 12

Name Clears the contents of the Exception 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_ClearERLogCmd_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

There are no error conditions for this command. If the Executive 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 dangerous. However, any previously logged data will be lost.

See also

CFE_ES_CLEAR_SYS_LOG_CC, CFE_ES_WRITE_SYS_LOG_CC, CFE_ES_WRITE_ER_LOG_CC

Definition at line 540 of file default_cfe_es_fcncodes.h.

12.102.2.2 CFE_ES_CLEAR_SYS_LOG_CC #define CFE_ES_CLEAR_SYS_LOG_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_ClearSysLogCmd_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

There are no error conditions for this command. If the Executive 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 dangerous. However, any previously logged data will be lost.

See also

[CFE_ES_WRITE_SYS_LOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#), [CFE_ES_WRITE_ER_LOG_CC](#), [CFE_ES_OVER_WRITE_SY](#)

Definition at line 463 of file default_cfe_es_fcncodes.h.

12.102.2.3 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_DeleteCDSCmd_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 specified CDS is the CDS portion of a Critical Table
- 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 909 of file default_cfe_es_fcncodes.h.

12.102.2.4 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_DumpCDSRegistryCmd_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 latest information.

Error Conditions

This command may fail for the following reason(s):

- The file name specified could not be parsed
- Error occurred while creating or writing to 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 990 of file default_cfe_es_fcncodes.h.

12.102.2.5 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_NoopCmd_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_BUILD_INF_EID](#) informational event message will be generated
- 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 73 of file default_cfe_es_fcncodes.h.

12.102.2.6 CFE_ES_OVER_WRITE_SYS_LOG_CC #define CFE_ES_OVER_WRITE_SYS_LOG_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_OverWriteSysLogCmd_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 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_SYS_LOG_CC](#), [CFE_ES_WRITE_SYS_LOG_CC](#)

Definition at line 792 of file default_cfe_es_fcncodes.h.

12.102.2.7 CFE_ES_QUERY_ALL_CC #define CFE_ES_QUERY_ALL_CC 9

Name Writes all Executive Services Information on all loaded modules to a File

Description

This command takes the information kept by Executive Services on all of the registered applications and libraries and writes it to the specified file.

Command Mnemonic(s) \$sc_\$cpu_ES_WriteAppInfo2File

Command Structure

[CFE_ES_QueryAllCmd_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 latest information.

Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- 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 428 of file `default_cfe_es_fcncodes.h`.

12.102.2.8 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_QueryAllTasksCmd_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 latest information.

Error Conditions

This command may fail for the following reason(s):

- The file name specified could not be parsed
- 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 1032 of file default_cfe_es_fcncodes.h.

12.102.2.9 CFE_ES_QUERY_ONE_CC #define CFE_ES_QUERY_ONE_CC 8

Name Request Executive Services Information on a specified module

Description

This command takes the information kept by Executive Services on the specified application or library and telemeters it to the ground.

Command Mnemonic(s) \$sc_\$cpu_ES_QueryApp

Command Structure

[CFE_ES_QueryOneCmd_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.
- Receipt of the [CFE_ES_OneAppTlm_t](#) telemetry packet

Error Conditions

This command may fail for the following reason(s):

- The specified name is not recognized as an active application or library

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 386 of file default_cfe_es_fcncodes.h.

12.102.2.10 CFE_ES_RELOAD_APP_CC #define CFE_ES_RELOAD_APP_CC 7

Name Stops, Unloads, Loads from the command specified 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_ReloadAppCmd_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 reload process has been initiated, not that it has completed.

Error Conditions

This command may fail for the following reason(s):

- The specified application filename string cannot be parsed
- 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 350 of file default_cfe_es_fcncodes.h.

12.102.2.11 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_ResetCountersCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_ES_CMDPC** - command execution counter and error counter will be reset to zero
- The [CFE_ES_RESET_INF_EID](#) informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Executive 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_ES_RESET_PR_COUNT_CC](#)

Definition at line 110 of file default_cfe_es_fcncodes.h.

12.102.2.12 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_ResetPRCn

Command Structure

[CFE_ES_ResetPRCountCmd_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

There are no error conditions for this command. If the Executive 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 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 829 of file default_cfe_es_fcncodes.h.

12.102.2.13 CFE_ES_RESTART_APP_CC #define CFE_ES_RESTART_APP_CC 6

Name Stops, Unloads, Loads using the previous File name, and Restarts an Application

Description

This command halts and removes the specified Application from the system. Then it immediately loads the Application from the same filename last used to start. 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_ResetApp

Command Structure

[CFE_ES_RestartAppCmd_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 restart process has been initiated, not that it has completed.

Error Conditions

This command may fail for the following reason(s):

- The original file is missing
- 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 304 of file default_cfe_es_fcncodes.h.

12.102.2.14 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_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_ES_ProcResetCnt** - processor reset counter will increment (processor reset) or reset to zero (power-on reset)
 - **\$sc_\$cpu_ES_ResetType** - processor reset type will be updated
 - **\$sc_\$cpu_ES_ResetSubtype** - processor reset subtype will be updated
 - 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 **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 162 of file default_cfe_es_fcncodes.h.

12.102.2.15 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_SendMemPoolStatsCmd_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 specified handle is not associated with a known memory pool

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 948 of file default_cfe_es_fcncodes.h.

12.102.2.16 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_SetMaxPRCntr

Command Structure

[CFE_ES_SetMaxPRCountCmd_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

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

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 867 of file default_cfe_es_fcncodes.h.

12.102.2.17 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_SetPerfFilterMaskCmd_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 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_MASK_CC](#)

Definition at line 715 of file default_cfe_es_fcncodes.h.

12.102.2.18 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_SetPerfTriggerMaskCmd_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 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 752 of file default_cfe_es_fcncodes.h.

12.102.2.19 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_StartAppCmd_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 specified application filename string cannot be parsed
- The specified application entry point is an empty string
- The specified application name is an empty string
- The specified priority is greater than 255
- The specified exception action is neither [CFE_ES_ExceptionAction_RESTART_APP](#) (0) or [CFE_ES_ExceptionAction_PROC](#) (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 205 of file default_cfe_es_fcncodes.h.

12.102.2.20 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_StartPerfDataCmd_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):

- 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_TRIGGER_MASK_CC](#)

Definition at line 628 of file default_cfe_es_fcncodes.h.

12.102.2.21 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_StopAppCmd_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 request has been initiated, not that it 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.
- `$sc_$cpu_ES_RegTasks` - number of tasks will decrease after tasks associated with app (main task and any child tasks) are stopped
- `$sc_$cpu_ES_RegExtApps` - external application counter will decrement after app is cleaned up

Error Conditions

This command may fail for the following reason(s):

- 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 258 of file default_cfe_es_fcncodes.h.

12.102.2.22 CFE_ES_STOP_PERF_DATA_CC #define CFE_ES_STOP_PERF_DATA_CC 15

Name Stop Performance Analyzer and write data file

Description

This command stops the Performance Analyzer from collecting any more data, and writes all previously collected performance data to a log file.

Command Mnemonic(s) \$sc_\$cpu_ES_StopLAData

Command Structure

[CFE_ES_StopPerfDataCmd_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 to indicate that data collection has been stopped. NOTE: Performance log data is written to the file as a background job. This event indicates that the file write process is initiated, not that it has completed.
- 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 latest information.

Error Conditions

This command may fail for the following reason(s):

- The file name specified could not be parsed
- Log data from a previous Stop Performance Analyzer command is still being written 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

NOTE: The performance analyzer data collection will still be stopped in the event of an error parsing the log file name or writing the log file.

Criticality

This command is not inherently dangerous. However, depending on configuration, performance data log files may be large in size and thus may fill the available storage.

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 678 of file `default_cfe_es_fcncodes.h`.

12.102.2.23 CFE_ES_WRITE_ER_LOG_CC #define CFE_ES_WRITE_ER_LOG_CC 13

Name Writes Exception 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_WriteERLogCmd_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_LOG_FILE](#) configuration parameter) will be updated with the latest information.

Error Conditions

This command may fail for the following reason(s):

- A previous request to write the ER log has not yet completed
- The specified FileName cannot be parsed
- 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_SYS_LOG_CC](#), [CFE_ES_WRITE_SYS_LOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#)

Definition at line 583 of file default_cfe_es_fcncodes.h.

12.102.2.24 CFE_ES_WRITE_SYS_LOG_CC #define CFE_ES_WRITE_SYS_LOG_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_WriteSysLogCmd_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 latest information.

Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- 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_SYS_LOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#), [CFE_ES_WRITE_ER_LOG_CC](#), [CFE_ES_OVER_WRITE_SYSLOG](#)

Definition at line 506 of file `default_cfe_es_fcncodes.h`.

12.103 cfe/modules/es/config/default_cfe_es_interface_cfg.h File Reference

Macros

- #define CFE_MISSION_ES_MAX_APPLICATIONS 16
- #define CFE_MISSION_ES_PERF_MAX_IDS 128
- #define CFE_MISSION_ES_POOL_MAX_BUCKETS 17
- #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
- #define CFE_MISSION_ES_DEFAULT_CRC CFE_ES_CrcType_16_ARC
- #define CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)

Checksum/CRC algorithm identifiers

- #define CFE_MISSION_ES_CRC_8 CFE_ES_CrcType_CRC_8
- #define CFE_MISSION_ES_CRC_16 CFE_ES_CrcType_CRC_16
- #define CFE_MISSION_ES_CRC_32 CFE_ES_CrcType_CRC_32

12.103.1 Detailed Description

CFE Executive Services (CFE_ES) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.103.2 Macro Definition Documentation

12.103.2.1 CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN #define CFE_MISSION_ES_CDS_MAX_FULL_NAME←
_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_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.CDSName"

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 143 of file default_cfe_es_interface_cfg.h.

12.103.2.2 CFE_MISSION_ES_CDS_MAX_NAME_LENGTH #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH←
TH 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"

This length does not need to include an extra character for NULL termination.

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 109 of file default_cfe_es_interface_cfg.h.

12.103.2.3 CFE_MISSION_ES_CRC_16 #define CFE_MISSION_ES_CRC_16 CFE_ES_CrcType_CRC_16

Definition at line 151 of file default_cfe_es_interface_cfg.h.

12.103.2.4 CFE_MISSION_ES_CRC_32 #define CFE_MISSION_ES_CRC_32 CFE_ES_CrcType_CRC_32
Definition at line 152 of file default_cfe_es_interface_cfg.h.

12.103.2.5 CFE_MISSION_ES_CRC_8 #define CFE_MISSION_ES_CRC_8 CFE_ES_CrcType_CRC_8
Definition at line 150 of file default_cfe_es_interface_cfg.h.

12.103.2.6 CFE_MISSION_ES_DEFAULT_CRC #define CFE_MISSION_ES_DEFAULT_CRC CFE_ES_CrcType_16_ARC

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_ES_CrcType_16_ARC is supported (see brief in CFE_ES_CrcType_Enum definition in [cfe_es_api_typedefs.h](#))

Definition at line 123 of file default_cfe_es_interface_cfg.h.

12.103.2.7 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 49 of file default_cfe_es_interface_cfg.h.

12.103.2.8 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.

Each performance id is used to identify something that needs to be measured. Performance ids are limited to the range of 0 to [CFE_MISSION_ES_PERF_MAX_IDS](#) - 1. Any performance ids outside of this range will be ignored and will be flagged as an error.

This affects the layout of 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 71 of file default_cfe_es_interface_cfg.h.

12.103.2.9 CFE_MISSION_ES_POOL_MAX_BUCKETS #define CFE_MISSION_ES_POOL_MAX_BUCKETS 17

Purpose Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS. This definition is used as the array size with the pool stats structure, and therefore should be consistent across all CPUs in a mission, as well as with the ground station.

There is also a platform-specific limit which may be fewer than this value.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

Definition at line 92 of file default_cfe_es_interface_cfg.h.

12.104 cfe/modules/es/config/default_cfe_es_internal_cfg.h File Reference**Macros**

- #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_ES_NONVOL_DISK_MOUNT_STRING "/cf"
- #define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"
- #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 256
- #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_CDS_SIZE (128 * 1024)
- #define CFE_PLATFORM_ES_USER_RESERVED_SIZE (1024 * 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_APP_LOG_FILE "/ram/cfe_es_app_info.log"
- #define CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE "/ram/cfe_es_taskinfo.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_POR_SYSLOG_MODE 0
- #define CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE 1
- #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_CDS_MAX_NUM_ENTRIES 512
- #define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
- #define CFE_PLATFORM_ES_POOL_MAX_BUCKETS 17
- #define CFE_PLATFORM_ES_MAX_MEMORY_POOLS 10
- #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_ES_STARTUP_SYNC_POLL_MSEC 50
- #define CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC 1000

12.104.1 Detailed Description

CFE Executive Services (CFE_ES) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.104.2 Macro Definition Documentation

12.104.2.1 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 responding and Calls it's RunLoop function, it will drop out of its 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 parameter. Units are number of [CFE_PLATFORM_ES_APP_SCAN_RATE](#) cycles.

Definition at line 232 of file default_cfe_es_internal_cfg.h.

12.104.2.2 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 parameter. millisecond units.

Definition at line 203 of file default_cfe_es_internal_cfg.h.

12.104.2.3 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE #define CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE 80000

Definition at line 774 of file default_cfe_es_internal_cfg.h.

12.104.2.4 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 664 of file default_cfe_es_internal_cfg.h.

12.104.2.5 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 758 of file default_cfe_es_internal_cfg.h.

12.104.2.6 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 16

Definition at line 759 of file default_cfe_es_internal_cfg.h.

12.104.2.7 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 32

Definition at line 760 of file default_cfe_es_internal_cfg.h.

12.104.2.8 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 48

Definition at line 761 of file default_cfe_es_internal_cfg.h.

12.104.2.9 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 64

Definition at line 762 of file default_cfe_es_internal_cfg.h.

12.104.2.10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 96

Definition at line 763 of file default_cfe_es_internal_cfg.h.

12.104.2.11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 128

Definition at line 764 of file default_cfe_es_internal_cfg.h.

12.104.2.12 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 160

Definition at line 765 of file default_cfe_es_internal_cfg.h.

12.104.2.13 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 256

Definition at line 766 of file default_cfe_es_internal_cfg.h.

12.104.2.14 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 512

Definition at line 767 of file default_cfe_es_internal_cfg.h.

12.104.2.15 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 1024

Definition at line 768 of file default_cfe_es_internal_cfg.h.

12.104.2.16 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 2048

Definition at line 769 of file default_cfe_es_internal_cfg.h.

12.104.2.17 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 4096

Definition at line 770 of file default_cfe_es_internal_cfg.h.

12.104.2.18 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK←
_SIZE_14 8192

Definition at line 771 of file default_cfe_es_internal_cfg.h.

12.104.2.19 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK←
_SIZE_15 16384

Definition at line 772 of file default_cfe_es_internal_cfg.h.

12.104.2.20 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK←
_SIZE_16 32768

Definition at line 773 of file default_cfe_es_internal_cfg.h.

12.104.2.21 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 parameter.

Definition at line 309 of file default_cfe_es_internal_cfg.h.

12.104.2.22 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 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 391 of file default_cfe_es_internal_cfg.h.

12.104.2.23 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 465 of file default_cfe_es_internal_cfg.h.

12.104.2.24 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 437 of file default_cfe_es_internal_cfg.h.

12.104.2.25 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 451 of file default_cfe_es_internal_cfg.h.

12.104.2.26 CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE #define CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE 0

Purpose Define Default System Log Mode following Power On Reset

Description:

Defines the default mode for the operation of the ES System log following a power on reset. 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. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 483 of file default_cfe_es_internal_cfg.h.

12.104.2.27 CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE #define CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE 1

Purpose Define Default System Log Mode following Processor Reset

Description:

Defines the default mode for the operation of the ES System log following a processor reset. 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. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 501 of file default_cfe_es_internal_cfg.h.

12.104.2.28 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 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 651 of file default_cfe_es_internal_cfg.h.

12.104.2.29 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_FI←
LE "/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 422 of file default_cfe_es_internal_cfg.h.

12.104.2.30 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_TASK_L←
OG_FILE "/ram/cfe_es_taskinfo.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 406 of file default_cfe_es_internal_cfg.h.

12.104.2.31 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 130 of file default_cfe_es_internal_cfg.h.

12.104.2.32 CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE #define CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE 256

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 144 of file default_cfe_es_internal_cfg.h.

12.104.2.33 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 103 of file default_cfe_es_internal_cfg.h.

12.104.2.34 CFE_PLATFORM_ES_MAX_BLOCK_SIZE #define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000

Definition at line 747 of file default_cfe_es_internal_cfg.h.

12.104.2.35 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 184 of file default_cfe_es_internal_cfg.h.

12.104.2.36 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 117 of file default_cfe_es_internal_cfg.h.

12.104.2.37 CFE_PLATFORM_ES_MAX_MEMORY_POOLS #define CFE_PLATFORM_ES_MAX_MEMORY_POOLS 10

Purpose Maximum number of memory pools

Description:

The upper limit for the number of memory pools that can concurrently exist within the system.

The CFE_SB and CFE_TBL core subsystems each define a memory pool.
Individual applications may also create memory pools, so this value should be set sufficiently high enough to support the applications being used on this platform.

Limits:

Must be at least 2 to support CFE core - SB and TBL pools. No specific upper limit.

Definition at line 712 of file default_cfe_es_internal_cfg.h.

12.104.2.38 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 679 of file default_cfe_es_internal_cfg.h.

12.104.2.39 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.

Definition at line 731 of file default_cfe_es_internal_cfg.h.

12.104.2.40 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 16
Definition at line 732 of file default_cfe_es_internal_cfg.h.

12.104.2.41 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 32
Definition at line 733 of file default_cfe_es_internal_cfg.h.

12.104.2.42 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 48
Definition at line 734 of file default_cfe_es_internal_cfg.h.

12.104.2.43 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 64
Definition at line 735 of file default_cfe_es_internal_cfg.h.

12.104.2.44 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 96
Definition at line 736 of file default_cfe_es_internal_cfg.h.

12.104.2.45 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 128
Definition at line 737 of file default_cfe_es_internal_cfg.h.

12.104.2.46 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 160
Definition at line 738 of file default_cfe_es_internal_cfg.h.

12.104.2.47 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
09 256

Definition at line 739 of file default_cfe_es_internal_cfg.h.

12.104.2.48 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
10 512

Definition at line 740 of file default_cfe_es_internal_cfg.h.

12.104.2.49 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
11 1024

Definition at line 741 of file default_cfe_es_internal_cfg.h.

12.104.2.50 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
12 2048

Definition at line 742 of file default_cfe_es_internal_cfg.h.

12.104.2.51 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
13 4096

Definition at line 743 of file default_cfe_es_internal_cfg.h.

12.104.2.52 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
14 8192

Definition at line 744 of file default_cfe_es_internal_cfg.h.

12.104.2.53 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
15 16384

Definition at line 745 of file default_cfe_es_internal_cfg.h.

12.104.2.54 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_↪
16 32768

Definition at line 746 of file default_cfe_es_internal_cfg.h.

12.104.2.55 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 348 of file default_cfe_es_internal_cfg.h.

12.104.2.56 CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING #define CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING "/cf"

Purpose Default virtual path for persistent storage

Description:

This configures the default location in the virtual file system for persistent/non-volatile storage. Files such as the startup script, app/library dynamic modules, and configuration tables are expected to be stored in this directory.

Definition at line 71 of file default_cfe_es_internal_cfg.h.

12.104.2.57 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 362 of file default_cfe_es_internal_cfg.h.

12.104.2.58 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 173 of file default_cfe_es_internal_cfg.h.

12.104.2.59 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 Performance 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 625 of file default_cfe_es_internal_cfg.h.

12.104.2.60 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 spawned 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 596 of file default_cfe_es_internal_cfg.h.

12.104.2.61 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 spawned 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 610 of file default_cfe_es_internal_cfg.h.

12.104.2.62 CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE #define CFE_PLATFORM_ES_PERF_DATA_BUFF←
ER_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 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 517 of file default_cfe_es_internal_cfg.h.

12.104.2.63 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 performance analyzer entries the Performance Analyzer Child Task will write to the file between delays.

Definition at line 635 of file default_cfe_es_internal_cfg.h.

12.104.2.64 CFE_PLATFORM_ES_PERF_FILTMASK_ALL #define CFE_PLATFORM_ES_PERF_FILTMASK_A←
LL ~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 537 of file default_cfe_es_internal_cfg.h.

12.104.2.65 CFE_PLATFORM_ES_PERF_FILTMASK_INIT #define CFE_PLATFORM_ES_PERF_FILTMASK_IN←
IT 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 548 of file default_cfe_es_internal_cfg.h.

12.104.2.66 CFE_PLATFORM_ES_PERF_FILTMASK_NONE #define CFE_PLATFORM_ES_PERF_FILTMASK_NONE↔
NE 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 527 of file default_cfe_es_internal_cfg.h.

12.104.2.67 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL #define CFE_PLATFORM_ES_PERF_TRIGMASK_ALL↔
LL ~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 570 of file default_cfe_es_internal_cfg.h.

12.104.2.68 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT #define CFE_PLATFORM_ES_PERF_TRIGMASK_INIT↔
IT 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 581 of file default_cfe_es_internal_cfg.h.

12.104.2.69 CFE_PLATFORM_ES_PERF_TRIGMASK_NONE #define CFE_PLATFORM_ES_PERF_TRIGMASK_NONE↔
NE 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 559 of file default_cfe_es_internal_cfg.h.

12.104.2.70 CFE_PLATFORM_ES_POOL_MAX_BUCKETS #define CFE_PLATFORM_ES_POOL_MAX_BUCKETS 17

Purpose Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

The ES and CDS block size lists must correlate with this value

Definition at line 694 of file default_cfe_es_internal_cfg.h.

12.104.2.71 CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING #define CFE_PLATFORM_ES_RAM_DISK_MOUN←
T_STRING "/ram"

Purpose Default virtual path for volatile storage

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 87 of file default_cfe_es_internal_cfg.h.

12.104.2.72 CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS #define CFE_PLATFORM_ES_RAM_DISK_NUM_S←
ECTORS 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 268 of file default_cfe_es_internal_cfg.h.

```
12.104.2.73 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 parameter. Units are percentage. A setting of zero will turn this feature off.

Definition at line 292 of file default_cfe_es_internal_cfg.h.

```
12.104.2.74 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 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 RAM disk sector size is system dependent and should be verified.

Definition at line 250 of file default_cfe_es_internal_cfg.h.

```
12.104.2.75 CFE_PLATFORM_ES_START_TASK_PRIORITY #define CFE_PLATFORM_ES_START_TASK_PRIORI←  
TY 68
```

Purpose Define ES Task Priority

Description:

Defines the cFE_ES Task priority.

Limits

Not Applicable

Definition at line 44 of file default_cfe_es_internal_cfg.h.

12.104.2.76 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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 59 of file default_cfe_es_internal_cfg.h.

12.104.2.77 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 allotted 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 814 of file default_cfe_es_internal_cfg.h.

12.104.2.78 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 796 of file default_cfe_es_internal_cfg.h.

12.104.2.79 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 159 of file default_cfe_es_internal_cfg.h.

12.104.2.80 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 parameter.

Definition at line 329 of file default_cfe_es_internal_cfg.h.

12.104.2.81 CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE #define CFE_PLATFORM_ES_VOLATILE_STARTUPFILE "/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 376 of file default_cfe_es_internal_cfg.h.

12.105 cfe/modules/es/config/default_cfe_es_mission_cfg.h File Reference

```
#include "cfe_es_interface_cfg.h"
```

12.105.1 Detailed Description

CFE Executive Services (CFE_ES) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.106 cfe/modules/es/config/default_cfe_es_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_es_fcncodes.h"
#include "cfe_es_msgdefs.h"
#include "cfe_es_msgstruct.h"
```

12.106.1 Detailed Description

Specification for the CFE Executive Services (CFE_ES) command and telemetry message data types.

This is a compatibility header for the "cfe_es_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.107 cfe/modules/es/config/default_cfe_es_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_es_fcncodes.h"
```

Data Structures

- struct [CFE_ES_RestartCmd_Payload](#)
Restart cFE Command Payload.
- struct [CFE_ES_FileNameCmd_Payload](#)
Generic file name command payload.
- struct [CFE_ES_OverWriteSysLogCmd_Payload](#)
Overwrite/Discard System Log Configuration Command Payload.
- struct [CFE_ES_StartAppCmd_Payload](#)
Start Application Command Payload.
- struct [CFE_ES_AppNameCmd_Payload](#)
Generic application name command payload.

- struct [CFE_ES_AppReloadCmd_Payload](#)
Reload Application Command Payload.
- struct [CFE_ES_SetMaxPRCountCmd_Payload](#)
Set Maximum Processor Reset Count Command Payload.
- struct [CFE_ES_DeleteCDSCmd_Payload](#)
Delete Critical Data Store Command Payload.
- struct [CFE_ES_StartPerfCmd_Payload](#)
Start Performance Analyzer Command Payload.
- struct [CFE_ES_StopPerfCmd_Payload](#)
Stop Performance Analyzer Command Payload.
- struct [CFE_ES_SetPerfFilterMaskCmd_Payload](#)
Set Performance Analyzer Filter Mask Command Payload.
- struct [CFE_ES_SetPerfTrigMaskCmd_Payload](#)
Set Performance Analyzer Trigger Mask Command Payload.
- struct [CFE_ES_SendMemPoolStatsCmd_Payload](#)
Send Memory Pool Statistics Command Payload.
- struct [CFE_ES_DumpCDSRegistryCmd_Payload](#)
Dump CDS Registry Command Payload.
- struct [CFE_ES_OneAppTlm_Payload](#)
- struct [CFE_ES_PoolStatsTlm_Payload](#)
- struct [CFE_ES_HousekeepingTlm_Payload](#)

Typedefs

- typedef struct [CFE_ES_RestartCmd_Payload](#) [CFE_ES_RestartCmd_Payload_t](#)
Restart cFE Command Payload.
- typedef struct [CFE_ES_FileNameCmd_Payload](#) [CFE_ES_FileNameCmd_Payload_t](#)
Generic file name command payload.
- typedef struct [CFE_ES_OverWriteSysLogCmd_Payload](#) [CFE_ES_OverWriteSysLogCmd_Payload_t](#)
Overwrite/Discard System Log Configuration Command Payload.
- typedef struct [CFE_ES_StartAppCmd_Payload](#) [CFE_ES_StartAppCmd_Payload_t](#)
Start Application Command Payload.
- typedef struct [CFE_ES_AppNameCmd_Payload](#) [CFE_ES_AppNameCmd_Payload_t](#)
Generic application name command payload.
- typedef struct [CFE_ES_AppReloadCmd_Payload](#) [CFE_ES_AppReloadCmd_Payload_t](#)
Reload Application Command Payload.
- typedef struct [CFE_ES_SetMaxPRCountCmd_Payload](#) [CFE_ES_SetMaxPRCountCmd_Payload_t](#)
Set Maximum Processor Reset Count Command Payload.
- typedef struct [CFE_ES_DeleteCDSCmd_Payload](#) [CFE_ES_DeleteCDSCmd_Payload_t](#)
Delete Critical Data Store Command Payload.
- typedef uint32 [CFE_ES_PerfMode_Enum_t](#)
- typedef struct [CFE_ES_StartPerfCmd_Payload](#) [CFE_ES_StartPerfCmd_Payload_t](#)
Start Performance Analyzer Command Payload.
- typedef struct [CFE_ES_StopPerfCmd_Payload](#) [CFE_ES_StopPerfCmd_Payload_t](#)
Stop Performance Analyzer Command Payload.
- typedef struct [CFE_ES_SetPerfFilterMaskCmd_Payload](#) [CFE_ES_SetPerfFilterMaskCmd_Payload_t](#)
Set Performance Analyzer Filter Mask Command Payload.
- typedef struct [CFE_ES_SetPerfTrigMaskCmd_Payload](#) [CFE_ES_SetPerfTrigMaskCmd_Payload_t](#)

Set Performance Analyzer Trigger Mask Command Payload.

- **typedef struct CFE_ES_SendMemPoolStatsCmd_Payload** `CFE_ES_SendMemPoolStatsCmd_Payload_t`
Send Memory Pool Statistics Command Payload.
- **typedef struct CFE_ES_DumpCDSRegistryCmd_Payload** `CFE_ES_DumpCDSRegistryCmd_Payload_t`
Dump CDS Registry Command Payload.
- **typedef struct CFE_ES_OneAppTlm_Payload** `CFE_ES_OneAppTlm_Payload_t`
- **typedef struct CFE_ES_PoolStatsTlm_Payload** `CFE_ES_PoolStatsTlm_Payload_t`
- **typedef struct CFE_ES_HousekeepingTlm_Payload** `CFE_ES_HousekeepingTlm_Payload_t`

Enumerations

- **enum CFE_ES_PerfMode** { `CFE_ES_PerfTrigger_START` = 0, `CFE_ES_PerfTrigger_CENTER`, `CFE_ES_PerfTrigger_END` }

Labels for values to use in `CFE_ES_StartPerfCmd_Payload.TriggerMode`.

12.107.1 Detailed Description

Specification for the CFE Executive Services (CFE_ES) command and telemetry message constant definitions.
For CFE_ES this is only the function/command code definitions

12.107.2 Typedef Documentation

12.107.2.1 CFE_ES_AppNameCmd_Payload_t `typedef struct CFE_ES_AppNameCmd_Payload` `CFE_ES_AppNameCmd_Payload_t`
Generic application name command payload.

For command details, see [CFE_ES_STOP_APP_CC](#), [CFE_ES_RESTART_APP_CC](#), [CFE_ES_QUERY_ONE_CC](#)

12.107.2.2 CFE_ES_AppReloadCmd_Payload_t `typedef struct CFE_ES_AppReloadCmd_Payload` `CFE_ES_AppReloadCmd_Payload_t`
Reload Application Command Payload.

For command details, see [CFE_ES_RELOAD_APP_CC](#)

12.107.2.3 CFE_ES_DeleteCDSCmd_Payload_t `typedef struct CFE_ES_DeleteCDSCmd_Payload` `CFE_ES_DeleteCDSCmd_Payload_t`
Delete Critical Data Store Command Payload.

For command details, see [CFE_ES_DELETE_CDS_CC](#)

12.107.2.4 CFE_ES_DumpCDSRegistryCmd_Payload_t `typedef struct CFE_ES_DumpCDSRegistryCmd_Payload` `CFE_ES_DumpCDSRegistryCmd_Payload_t`
Dump CDS Registry Command Payload.

For command details, see [CFE_ES_DUMP_CDS_REGISTRY_CC](#)

12.107.2.5 CFE_ES_FileNameCmd_Payload_t `typedef struct CFE_ES_FileNameCmd_Payload` `CFE_ES_FileNameCmd_Payload_t`
Generic file name command payload.

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_SYS_LOG_CC](#), and [CFE_ES_WRITE_ER_LOG_CC](#)

12.107.2.6 CFE_ES_HousekeepingTlm_Payload_t `typedef struct CFE_ES_HousekeepingTlm_Payload` `CFE_ES_HousekeepingTlm_Payload_t`

Name Executive Services Housekeeping Packet

12.107.2.7 CFE_ES_OneAppTlm_Payload_t `typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t`

Name Single Application Information Packet

12.107.2.8 CFE_ES_OverWriteSysLogCmd_Payload_t `typedef struct CFE_ES_OverWriteSysLogCmd_Payload CFE_ES_OverWriteSysLogCmd_Payload_t`

Overwrite/Discard System Log Configuration Command Payload.

For command details, see [CFE_ES_OVER_WRITE_SYS_LOG_CC](#)

12.107.2.9 CFE_ES_PerfMode_Enum_t `typedef uint32 CFE_ES_PerfMode_Enum_t`

Definition at line 157 of file default_cfe_es_msgdefs.h.

12.107.2.10 CFE_ES_PoolStatsTlm_Payload_t `typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t`

Name Memory Pool Statistics Packet

12.107.2.11 CFE_ES_RestartCmd_Payload_t `typedef struct CFE_ES_RestartCmd_Payload CFE_ES_RestartCmd_Payload_t`

Restart cFE Command Payload.

For command details, see [CFE_ES_RESTART_CC](#)

12.107.2.12 CFE_ES_SendMemPoolStatsCmd_Payload_t `typedef struct CFE_ES_SendMemPoolStatsCmd_Payload CFE_ES_SendMemPoolStatsCmd_Payload_t`

Send Memory Pool Statistics Command Payload.

For command details, see [CFE_ES_SEND_MEM_POOL_STATS_CC](#)

12.107.2.13 CFE_ES_SetMaxPRCountCmd_Payload_t `typedef struct CFE_ES_SetMaxPRCountCmd_Payload CFE_ES_SetMaxPRCountCmd_Payload_t`

Set Maximum Processor Reset Count Command Payload.

For command details, see [CFE_ES_SET_MAX_PR_COUNT_CC](#)

12.107.2.14 CFE_ES_SetPerfFilterMaskCmd_Payload_t `typedef struct CFE_ES_SetPerfFilterMaskCmd_Payload CFE_ES_SetPerfFilterMaskCmd_Payload_t`

Set Performance Analyzer Filter Mask Command Payload.

For command details, see [CFE_ES_SET_PERF_FILTER_MASK_CC](#)

12.107.2.15 CFE_ES_SetPerfTrigMaskCmd_Payload_t `typedef struct CFE_ES_SetPerfTrigMaskCmd_Payload CFE_ES_SetPerfTrigMaskCmd_Payload_t`

Set Performance Analyzer Trigger Mask Command Payload.

For command details, see [CFE_ES_SET_PERF_TRIGGER_MASK_CC](#)

12.107.2.16 CFE_ES_StartAppCmd_Payload_t `typedef struct CFE_ES_StartAppCmd_Payload CFE_ES_StartAppCmd_Payload_t`

Start Application Command Payload.

For command details, see [CFE_ES_START_APP_CC](#)

12.107.2.17 CFE_ES_StartPerfCmd_Payload_t `typedef struct CFE_ES_StartPerfCmd_Payload CFE_ES_StartPerfCmd_Payload_t`

Start Performance Analyzer Command Payload.

For command details, see [CFE_ES_START_PERF_DATA_CC](#)

12.107.2.18 CFE_ES_StopPerfCmd_Payload_t `typedef struct CFE_ES_StopPerfCmd_Payload CFE_ES_StopPerfCmd_Payload_t`
Stop Performance Analyzer Command Payload.
For command details, see [CFE_ES_STOP_PERF_DATA_CC](#)

12.107.3 Enumeration Type Documentation

12.107.3.1 CFE_ES_PerfMode `enum CFE_ES_PerfMode`
Labels for values to use in [CFE_ES_StartPerfCmd_Payload.TriggerMode](#).

See also

[CFE_ES_StartPerfCmd_Payload](#)

Enumerator

<code>CFE_ES_PerfTrigger_START</code>	
<code>CFE_ES_PerfTrigger_CENTER</code>	
<code>CFE_ES_PerfTrigger_END</code>	

Definition at line 150 of file default_cfe_es_msgdefs.h.

12.108 cfe/modules/es/config/default_cfe_es_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_es_topicids.h"
```

Macros

- `#define CFE_ES_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_ES_CMD_TOPICID)
/* 0x1806 */`
- `#define CFE_ES_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_ES_SEND_HK_TOPICID)
/* 0x1808 */`
- `#define CFE_ES_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_HK_TLM_TOPICID)
/* 0x0800 */`
- `#define CFE_ES_APP_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_APP_TLM_TOPICID)
/* 0x080B */`
- `#define CFE_ES_MEMSTATS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_MEMSTATS_TLM_TOPICID)
/* 0x0810 */`

12.108.1 Detailed Description

CFE Executive Services (CFE_ES) Application Message IDs

12.108.2 Macro Definition Documentation

12.108.2.1 CFE_ES_APP_TLM_MID `#define CFE_ES_APP_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_APP_TLM_TOPICID)`
/* 0x080B */

Definition at line 39 of file default_cfe_es_msgids.h.

12.108.2.2 CFE_ES_CMD_MID #define CFE_ES_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_ES_CMD_TOPICID)
/* 0x1806 */
Definition at line 32 of file default_cfe_es_msgids.h.

12.108.2.3 CFE_ES_HK_TLM_MID #define CFE_ES_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_HK_TLM)
/* 0x0800 */
Definition at line 38 of file default_cfe_es_msgids.h.

12.108.2.4 CFE_ES_MEMSTATS_TLM_MID #define CFE_ES_MEMSTATS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_ES_MEMSTATS)
/* 0x0810 */
Definition at line 40 of file default_cfe_es_msgids.h.

12.108.2.5 CFE_ES_SEND_HK_MID #define CFE_ES_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_ES_SEND_HK)
/* 0x1808 */
Definition at line 33 of file default_cfe_es_msgids.h.

12.109 cfe/modules/es/config/default_cfe_es_msgstruct.h File Reference

```
#include "cfe_es_msgdefs.h"
#include "cfe_msg_hdr.h"
#include "cfe_mission_cfg.h"
```

Data Structures

- struct **CFE_ES_NoopCmd**
- struct **CFE_ES_ResetCountersCmd**
- struct **CFE_ES_ClearSysLogCmd**
- struct **CFE_ES_ClearERLogCmd**
- struct **CFE_ES_ResetPRCountCmd**
- struct **CFE_ES_SendHkCmd**
- struct **CFE_ES_RestartCmd**
Restart cFE Command.
- struct **CFE_ES_FileNameCmd**
Generic file name command.
- struct **CFE_ES_QueryAllCmd**
- struct **CFE_ES_QueryAllTasksCmd**
- struct **CFE_ES_WriteSysLogCmd**
- struct **CFE_ES_WriteERLogCmd**
- struct **CFE_ES_OverWriteSysLogCmd**
Overwrite/Discard System Log Configuration Command Payload.
- struct **CFE_ES_StartApp**
Start Application Command.
- struct **CFE_ES_StopAppCmd**
- struct **CFE_ES_RestartAppCmd**
- struct **CFE_ES_QueryOneCmd**
- struct **CFE_ES_ReloadAppCmd**
Reload Application Command.

- struct [CFE_ES_SetMaxPRCountCmd](#)
Set Maximum Processor Reset Count Command.
- struct [CFE_ES_DeleteCDSCmd](#)
Delete Critical Data Store Command.
- struct [CFE_ES_StartPerfDataCmd](#)
Start Performance Analyzer Command.
- struct [CFE_ES_StopPerfDataCmd](#)
Stop Performance Analyzer Command.
- struct [CFE_ES_SetPerfFilterMaskCmd](#)
Set Performance Analyzer Filter Mask Command.
- struct [CFE_ES_SetPerfTriggerMaskCmd](#)
Set Performance Analyzer Trigger Mask Command.
- struct [CFE_ES_SendMemPoolStatsCmd](#)
Send Memory Pool Statistics Command.
- struct [CFE_ES_DumpCDSRegistryCmd](#)
Dump CDS Registry Command.
- struct [CFE_ES_OneAppTlm](#)
- struct [CFE_ES_MemStatsTlm](#)
- struct [CFE_ES_HousekeepingTlm](#)

Typedefs

- typedef struct [CFE_ES_NoopCmd](#) [CFE_ES_NoopCmd_t](#)
- typedef struct [CFE_ES_ResetCountersCmd](#) [CFE_ES_ResetCountersCmd_t](#)
- typedef struct [CFE_ES_ClearSysLogCmd](#) [CFE_ES_ClearSysLogCmd_t](#)
- typedef struct [CFE_ES_ClearERLogCmd](#) [CFE_ES_ClearERLogCmd_t](#)
- typedef struct [CFE_ES_ResetPRCountCmd](#) [CFE_ES_ResetPRCountCmd_t](#)
- typedef struct [CFE_ES_SendHkCmd](#) [CFE_ES_SendHkCmd_t](#)
- typedef struct [CFE_ES_RestartCmd](#) [CFE_ES_RestartCmd_t](#)
Restart cFE Command.
- typedef struct [CFE_ES_FileNameCmd](#) [CFE_ES_FileNameCmd_t](#)
Generic file name command.
- typedef struct [CFE_ES_QueryAllCmd](#) [CFE_ES_QueryAllCmd_t](#)
- typedef struct [CFE_ES_QueryAllTasksCmd](#) [CFE_ES_QueryAllTasksCmd_t](#)
- typedef struct [CFE_ES_WriteSysLogCmd](#) [CFE_ES_WriteSysLogCmd_t](#)
- typedef struct [CFE_ES_WriteERLogCmd](#) [CFE_ES_WriteERLogCmd_t](#)
- typedef struct [CFE_ES_OverWriteSysLogCmd](#) [CFE_ES_OverWriteSysLogCmd_t](#)
Overwrite/Discard System Log Configuration Command Payload.
- typedef struct [CFE_ES_StartApp](#) [CFE_ES_StartAppCmd_t](#)
Start Application Command.
- typedef struct [CFE_ES_StopAppCmd](#) [CFE_ES_StopAppCmd_t](#)
- typedef struct [CFE_ES_RestartAppCmd](#) [CFE_ES_RestartAppCmd_t](#)
- typedef struct [CFE_ES_QueryOneCmd](#) [CFE_ES_QueryOneCmd_t](#)
- typedef struct [CFE_ES_ReloadAppCmd](#) [CFE_ES_ReloadAppCmd_t](#)
Reload Application Command.
- typedef struct [CFE_ES_SetMaxPRCountCmd](#) [CFE_ES_SetMaxPRCountCmd_t](#)
Set Maximum Processor Reset Count Command.
- typedef struct [CFE_ES_DeleteCDSCmd](#) [CFE_ES_DeleteCDSCmd_t](#)
Delete Critical Data Store Command.

- **typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t**
Start Performance Analyzer Command.
- **typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t**
Stop Performance Analyzer Command.
- **typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t**
Set Performance Analyzer Filter Mask Command.
- **typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t**
Set Performance Analyzer Trigger Mask Command.
- **typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t**
Send Memory Pool Statistics Command.
- **typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t**
Dump CDS Registry Command.
- **typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t**
- **typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t**
- **typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t**

12.109.1 Detailed Description

Purpose: cFE Executive Services (ES) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

12.109.2 Typedef Documentation

12.109.2.1 CFE_ES_ClearERLogCmd_t `typedef struct CFE_ES_ClearERLogCmd CFE_ES_ClearERLogCmd_t`

12.109.2.2 CFE_ES_ClearSysLogCmd_t `typedef struct CFE_ES_ClearSysLogCmd CFE_ES_ClearSysLogCmd_t`

12.109.2.3 CFE_ES_DeleteCDSCmd_t `typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t`
Delete Critical Data Store Command.

12.109.2.4 CFE_ES_DumpCDSRegistryCmd_t `typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t`
Dump CDS Registry Command.

12.109.2.5 CFE_ES_FileNameCmd_t `typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t`
Generic file name command.

12.109.2.6 CFE_ES_HousekeepingTlm_t `typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t`

Name Executive Services Housekeeping Packet

12.109.2.7 CFE_ES_MemStatsTlm_t `typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t`

Name Memory Pool Statistics Packet

12.109.2.8 CFE_ES_NoopCmd_t `typedef struct CFE_ES_NoopCmd CFE_ES_NoopCmd_t`

12.109.2.9 CFE_ES_OneAppTlm_t `typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t`

Name Single Application Information Packet

12.109.2.10 CFE_ES_OverWriteSysLogCmd_t `typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_t`
Overwrite/Discard System Log Configuration Command Payload.

12.109.2.11 CFE_ES_QueryAllCmd_t `typedef struct CFE_ES_QueryAllCmd CFE_ES_QueryAllCmd_t`

12.109.2.12 CFE_ES_QueryAllTasksCmd_t `typedef struct CFE_ES_QueryAllTasksCmd CFE_ES_QueryAllTasksCmd_t`

12.109.2.13 CFE_ES_QueryOneCmd_t `typedef struct CFE_ES_QueryOneCmd CFE_ES_QueryOneCmd_t`

12.109.2.14 CFE_ES_ReloadAppCmd_t `typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t`
Reload Application Command.

12.109.2.15 CFE_ES_ResetCountersCmd_t `typedef struct CFE_ES_ResetCountersCmd CFE_ES_ResetCountersCmd_t`

12.109.2.16 CFE_ES_ResetPRCountCmd_t `typedef struct CFE_ES_ResetPRCountCmd CFE_ES_ResetPRCountCmd_t`

12.109.2.17 CFE_ES_RestartAppCmd_t `typedef struct CFE_ES_RestartAppCmd CFE_ES_RestartAppCmd_t`

12.109.2.18 CFE_ES_RestartCmd_t `typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t`
Restart cFE Command.

12.109.2.19 CFE_ES_SendHkCmd_t `typedef struct CFE_ES_SendHkCmd CFE_ES_SendHkCmd_t`

12.109.2.20 CFE_ES_SendMemPoolStatsCmd_t `typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t`
Send Memory Pool Statistics Command.

12.109.2.21 CFE_ES_SetMaxPRCountCmd_t `typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t`
Set Maximum Processor Reset Count Command.

12.109.2.22 CFE_ES_SetPerfFilterMaskCmd_t `typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t`
Set Performance Analyzer Filter Mask Command.

12.109.2.23 CFE_ES_SetPerfTriggerMaskCmd_t `typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t`
Set Performance Analyzer Trigger Mask Command.

12.109.2.24 CFE_ES_StartAppCmd_t `typedef struct CFE_ES_StartApp CFE_ES_StartAppCmd_t`
Start Application Command.

12.109.2.25 CFE_ES_StartPerfDataCmd_t `typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t`
Start Performance Analyzer Command.

12.109.2.26 CFE_ES_StopAppCmd_t `typedef struct CFE_ES_StopAppCmd CFE_ES_StopAppCmd_t`

12.109.2.27 CFE_ES_StopPerfDataCmd_t `typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t`
Stop Performance Analyzer Command.

12.109.2.28 CFE_ES_WriteERLogCmd_t `typedef struct CFE_ES_WriteERLogCmd CFE_ES_WriteERLogCmd_t`

12.109.2.29 CFE_ES_WriteSysLogCmd_t `typedef struct CFE_ES_WriteSysLogCmd CFE_ES_WriteSysLogCmd_t`

12.110 cfe/modules/es/config/default_cfe_es_platform_cfg.h File Reference

```
#include "cfe_es_mission_cfg.h"
#include "cfe_es_internal_cfg.h"
```

12.110.1 Detailed Description

CFE Executive Services (CFE_ES) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.111 cfe/modules/es/config/default_cfe_es_topicids.h File Reference

Macros

- #define CFE_MISSION_ES_CMD_TOPICID 6
- #define CFE_MISSION_ES_SEND_HK_TOPICID 8
- #define CFE_MISSION_ES_HK_TLM_TOPICID 0
- #define CFE_MISSION_ES_APP_TLM_TOPICID 11
- #define CFE_MISSION_ES_MEMSTATS_TLM_TOPICID 16

12.111.1 Detailed Description

CFE Executive Services (CFE_ES) Application Topic IDs

12.111.2 Macro Definition Documentation

12.111.2.1 CFE_MISSION_ES_APP_TLM_TOPICID #define CFE_MISSION_ES_APP_TLM_TOPICID 11
Definition at line 48 of file default_cfe_es_topicids.h.

12.111.2.2 CFE_MISSION_ES_CMD_TOPICID #define CFE_MISSION_ES_CMD_TOPICID 6

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE ES command messages

Limits

Not Applicable

Definition at line 35 of file default_cfe_es_topicids.h.

12.111.2.3 CFE_MISSION_ES_HK_TLM_TOPICID #define CFE_MISSION_ES_HK_TLM_TOPICID 0

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE ES telemetry messages

Limits

Not Applicable

Definition at line 47 of file default_cfe_es_topicids.h.

12.111.2.4 CFE_MISSION_ES_MEMSTATS_TLM_TOPICID #define CFE_MISSION_ES_MEMSTATS_TLM_TOPICID 16
Definition at line 49 of file default_cfe_es_topicids.h.

12.111.2.5 CFE_MISSION_ES_SEND_HK_TOPICID #define CFE_MISSION_ES_SEND_HK_TOPICID 8
Definition at line 36 of file default_cfe_es_topicids.h.

12.112 cfe/modules/es/fsw/inc/cfe_es_eventids.h File Reference

Macros

ES event IDs

- #define CFE_ES_INIT_INF_EID 1
ES Initialization Event ID.
- #define CFE_ES_INITSTATS_INF_EID 2
ES Initialization Statistics Information Event ID.
- #define CFE_ES_NOOP_INF_EID 3
ES No-op Command Success Event ID.
- #define CFE_ES_RESET_INF_EID 4
ES Reset Counters Command Success Event ID.
- #define CFE_ES_START_INF_EID 6
ES Start Application Command Success Event ID.
- #define CFE_ES_STOP_DBG_EID 7
ES Stop Application Command Request Success Event ID.
- #define CFE_ES_STOP_INF_EID 8
ES Stop Application Completed Event ID.
- #define CFE_ES_RESTART_APP_DBG_EID 9
ES Restart Application Command Request Success Event ID.
- #define CFE_ES_RESTART_APP_INF_EID 10
ES Restart Application Completed Event ID.
- #define CFE_ES_RELOAD_APP_DBG_EID 11
ES Reload Application Command Request Success Event ID.
- #define CFE_ES_RELOAD_APP_INF_EID 12
ES Reload Application Complete Event ID.
- #define CFE_ES_EXIT_APP_INF_EID 13
ES Nominal Exit Application Complete Event ID.
- #define CFE_ES_ERREXIT_APP_INF_EID 14
ES Error Exit Application Complete Event ID.
- #define CFE_ES_ONE_APP_EID 15
ES Query One Application Command Success Event ID.
- #define CFE_ES_ALL_APPS_EID 16
ES Query All Applications Command Success Event ID.
- #define CFE_ES_SYSLOG1_INF_EID 17
ES Clear System Log Command Success Event ID.
- #define CFE_ES_SYSLOG2_EID 18
ES Write System Log Command Success Event ID.
- #define CFE_ES_ERLOG1_INF_EID 19
ES Clear Exception Reset Log Command Success Event ID.
- #define CFE_ES_ERLOG2_EID 20
ES Write Exception Reset Log Complete Event ID.
- #define CFE_ES_MID_ERR_EID 21
ES Invalid Message ID Received Event ID.
- #define CFE_ES_CC1_ERR_EID 22
ES Invalid Command Code Received Event ID.
- #define CFE_ES_LEN_ERR_EID 23
ES Invalid Command Length Event ID.
- #define CFE_ES_BOOT_ERR_EID 24
ES Restart Command Invalid Restart Type Event ID.

- #define `CFE_ES_START_ERR_EID` 26
ES Start Application Command Application Creation Failed Event ID.
- #define `CFE_ES_START_INVALID_FILENAME_ERR_EID` 27
ES Start Application Command Invalid Filename Event ID.
- #define `CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID` 28
ES Start Application Command Entry Point NULL Event ID.
- #define `CFE_ES_START_NULL_APP_NAME_ERR_EID` 29
ES Start Application Command App Name NULL Event ID.
- #define `CFE_ES_START_PRIORITY_ERR_EID` 31
ES Start Application Command Priority Too Large Event ID.
- #define `CFE_ES_START_EXC_ACTION_ERR_EID` 32
ES Start Application Command Exception Action Invalid Event ID.
- #define `CFE_ES_ERREXIT_APP_ERR_EID` 33
ES Error Exit Application Cleanup Failed Event ID.
- #define `CFE_ES_STOP_ERR1_EID` 35
ES Stop Application Command Request Failed Event ID.
- #define `CFE_ES_STOP_ERR2_EID` 36
ES Stop Application Command Get AppID By Name Failed Event ID.
- #define `CFE_ES_STOP_ERR3_EID` 37
ES Stop Application Cleanup Failed Event ID.
- #define `CFE_ES_RESTART_APP_ERR1_EID` 38
ES Restart Application Command Request Failed Event ID.
- #define `CFE_ES_RESTART_APP_ERR2_EID` 39
ES Restart Application Command Get AppID By Name Failed Event ID.
- #define `CFE_ES_RESTART_APP_ERR3_EID` 40
ES Restart Application Startup Failed Event ID.
- #define `CFE_ES_RESTART_APP_ERR4_EID` 41
ES Restart Application Cleanup Failed Event ID.
- #define `CFE_ES_RELOAD_APP_ERR1_EID` 42
ES Reload Application Command Request Failed Event ID.
- #define `CFE_ES_RELOAD_APP_ERR2_EID` 43
ES Reload Application Command Get AppID By Name Failed Event ID.
- #define `CFE_ES_RELOAD_APP_ERR3_EID` 44
ES Reload Application Startup Failed Event ID.
- #define `CFE_ES_RELOAD_APP_ERR4_EID` 45
ES Reload Application Cleanup Failed Event ID.
- #define `CFE_ES_EXIT_APP_ERR_EID` 46
ES Exit Application Cleanup Failed Event ID.
- #define `CFE_ES_PCR_ERR1_EID` 47
ES Process Control Invalid Exception State Event ID.
- #define `CFE_ES_PCR_ERR2_EID` 48
ES Process Control Unknown State Event ID.
- #define `CFE_ES_ONE_ERR_EID` 49
ES Query One Application Data Command Transmit Message Failed Event ID.
- #define `CFE_ES_ONE_APPID_ERR_EID` 50
ES Query One Application Data Command Get AppID By Name Failed Event ID.
- #define `CFE_ES_OSCREATE_ERR_EID` 51
ES Query All Application Data Command File Creation Failed Event ID.
- #define `CFE_ES_WRHDR_ERR_EID` 52
ES Query All Application Data Command File Write Header Failed Event ID.
- #define `CFE_ES_TASKWR_ERR_EID` 53
ES Query All Application Data Command File Write App Data Failed Event ID.
- #define `CFE_ES_SYSLOG2_ERR_EID` 55
ES Write System Log Command Filename Parse or File Creation Failed Event ID.
- #define `CFE_ES_ERLOG2_ERR_EID` 56

- #define `CFE_ES_PERF_STARTCMD_EID` 57
ES Start Performance Analyzer Data Collection Command Success Event ID.
- #define `CFE_ES_PERF_STARTCMD_ERR_EID` 58
ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.
- #define `CFE_ES_PERF_STARTCMD_TRIG_ERR_EID` 59
ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.
- #define `CFE_ES_PERF_STOPCMD_EID` 60
ES Stop Performance Analyzer Data Collection Command Request Success Event ID.
- #define `CFE_ES_PERF_STOPCMD_ERR2_EID` 62
ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.
- #define `CFE_ES_PERF_FILTMSKCMD_EID` 63
ES Set Performance Analyzer Filter Mask Command Success Event ID.
- #define `CFE_ES_PERF_FILTMSKERR_EID` 64
ES Set Performance Analyzer Filter Mask Command Invalid Index Event ID.
- #define `CFE_ES_PERF_TRIGMSKCMD_EID` 65
ES Set Performance Analyzer Trigger Mask Command Success Event ID.
- #define `CFE_ES_PERF_TRIGMSKERR_EID` 66
ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.
- #define `CFE_ES_PERF_LOG_ERR_EID` 67
ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.
- #define `CFE_ES_PERF_DATAWRITTEN_EID` 68
Performance Log Write Success Event ID.
- #define `CFE_ES_CDS_REGISTER_ERR_EID` 69
ES Register CDS API Failed Event ID.
- #define `CFE_ES_SYSLOGMODE_EID` 70
ES Set System Log Overwrite Mode Command Success Event ID.
- #define `CFE_ES_ERR_SYSLOGMODE_EID` 71
ES Set System Log Overwrite Mode Command Failed Event ID.
- #define `CFE_ES_RESET_PR_COUNT_EID` 72
ES Set Processor Reset Counter to Zero Command Success Event ID.
- #define `CFE_ES_SET_MAX_PR_COUNT_EID` 73
ES Set Maximum Processor Reset Limit Command Success Event ID.
- #define `CFE_ES_FILEWRITE_ERR_EID` 74
ES File Write Failed Event ID.
- #define `CFE_ES_CDS_DELETE_ERR_EID` 76
ES Delete CDS Command Delete Failed Event ID.
- #define `CFE_ES_CDS_NAME_ERR_EID` 77
ES Delete CDS Command Lookup CDS Failed Event ID.
- #define `CFE_ES_CDS_DELETED_INFO_EID` 78
ES Delete CDS Command Success Event ID.
- #define `CFE_ES_CDS_DELETE_TBL_ERR_EID` 79
ES Delete CDS Command For Critical Table Event ID.
- #define `CFE_ES_CDS_OWNER_ACTIVE_EID` 80
ES Delete CDS Command With Active Owner Event ID.
- #define `CFE_ES_TLM_POOL_STATS_INFO_EID` 81
ES Telemeter Memory Statistics Command Success Event ID.
- #define `CFE_ES_INVALID_POOL_HANDLE_ERR_EID` 82
ES Telemeter Memory Statistics Command Invalid Handle Event ID.
- #define `CFE_ES_CDS_REG_DUMP_INF_EID` 83
ES Write Critical Data Store Registry Command Success Event ID.
- #define `CFE_ES_CDS_DUMP_ERR_EID` 84
ES Write Critical Data Store Registry Command Record Write Failed Event ID.
- #define `CFE_ES_WRITE_CFE_HDR_ERR_EID` 85
ES Write Critical Data Store Registry Command Header Write Failed Event ID.

- #define [CFE_ES_CREATING_CDS_DUMP_ERR_EID](#) 86
ES Write Critical Data Store Registry Command Filename Parse or File Create Failed Event ID.
- #define [CFE_ES_TASKINFO_EID](#) 87
ES Write All Task Data Command Success Event ID.
- #define [CFE_ES_TASKINFO_OSCREATE_ERR_EID](#) 88
ES Write All Task Data Command Filename Parse or File Create Failed Event ID.
- #define [CFE_ES_TASKINFO_WRHDR_ERR_EID](#) 89
ES Write All Task Data Command Write Header Failed Event ID.
- #define [CFE_ES_TASKINFO_WR_ERR_EID](#) 90
ES Write All Task Data Command Write Data Failed Event ID.
- #define [CFE_ES_VERSION_INF_EID](#) 91
CFS Version Information Event ID
- #define [CFE_ES_BUILD_INF_EID](#) 92
CFS Build Information Event ID
- #define [CFE_ES_ERLOG_PENDING_ERR_EID](#) 93
ES Write Exception Reset Log Command Already In Progress Event ID.

12.112.1 Detailed Description

cFE Executive Services Event IDs

12.112.2 Macro Definition Documentation

12.112.2.1 CFE_ES_ALL_APPS_EID #define [CFE_ES_ALL_APPS_EID](#) 16
ES Query All Applications Command Success Event ID.

Type: DEBUG

Cause:

[ES Query All Applications Command](#) success.
Definition at line 206 of file cfe_es_eventids.h.

12.112.2.2 CFE_ES_BOOT_ERR_EID #define [CFE_ES_BOOT_ERR_EID](#) 24
ES Restart Command Invalid Restart Type Event ID.

Type: ERROR

Cause:

[ES cFE Restart Command](#) failure due to invalid restart type.
Definition at line 294 of file cfe_es_eventids.h.

12.112.2.3 CFE_ES_BUILD_INF_EID #define CFE_ES_BUILD_INF_EID 92
cFS Build Information Event ID

Type: INFORMATION

Cause:

ES Initialization complete and response to [ES NO-OP 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"

This additionally reports the configuration name that was selected by the user, which may affect various platform/mission limits.

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 1047 of file cfe_es_eventids.h.

12.112.2.4 CFE_ES_CC1_ERR_EID #define CFE_ES_CC1_ERR_EID 22
ES Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE_ES_CMD_MID](#) received on the ES message pipe.

Definition at line 272 of file cfe_es_eventids.h.

12.112.2.5 CFE_ES_CDS_DELETE_ERR_EID #define CFE_ES_CDS_DELETE_ERR_EID 76
ES Delete CDS Command Delete Failed Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failed while deleting, see reported status code or system log for details.

Definition at line 834 of file cfe_es_eventids.h.

12.112.2.6 CFE_ES_CDS_DELETE_TBL_ERR_EID #define CFE_ES_CDS_DELETE_TBL_ERR_EID 79
ES Delete CDS Command For Critical Table Event ID.

Type: ERROR

Cause:

[Delete CDS Command](#) failure due to the specified CDS name being a critical table. Critical Table images can only be deleted via a Table Services command, [CFE_TBL_DELETE_CDS_CC](#).

Definition at line 871 of file cfe_es_eventids.h.

12.112.2.7 CFE_ES_CDS_DELETED_INFO_EID #define CFE_ES_CDS_DELETED_INFO_EID 78
ES Delete CDS Command Success Event ID.

Type: INFORMATION

Cause:

[ES Delete CDS Command](#) success.

Definition at line 857 of file cfe_es_eventids.h.

12.112.2.8 CFE_ES_CDS_DUMP_ERR_EID #define CFE_ES_CDS_DUMP_ERR_EID 84
ES Write Critical Data Store Registry Command Record Write Failed Event ID.

Type: ERROR

Cause:

[ES Write Critical Data Store Registry Command](#) failed to write CDS record.

Definition at line 929 of file cfe_es_eventids.h.

12.112.2.9 CFE_ES_CDS_NAME_ERR_EID #define CFE_ES_CDS_NAME_ERR_EID 77
ES Delete CDS Command Lookup CDS Failed Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failed due to the specified CDS name not found in the CDS Registry.

Definition at line 846 of file cfe_es_eventids.h.

12.112.2.10 CFE_ES_CDS_OWNER_ACTIVE_EID #define CFE_ES_CDS_OWNER_ACTIVE_EID 80
ES Delete CDS Command With Active Owner Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failure due to the specifies CDS name is registered to an active application.
Definition at line 883 of file cfe_es_eventids.h.

12.112.2.11 CFE_ES_CDS_REG_DUMP_INF_EID #define CFE_ES_CDS_REG_DUMP_INF_EID 83
ES Write Critical Data Store Registry Command Success Event ID.

Type: DEBUG

Cause:

[ES Write Critical Data Store Registry Command](#) success.
Definition at line 917 of file cfe_es_eventids.h.

12.112.2.12 CFE_ES_CDS_REGISTER_ERR_EID #define CFE_ES_CDS_REGISTER_ERR_EID 69
ES Register CDS API Failed Event ID.

Type: ERROR

Cause:

[CFE_ES_RegisterCDS](#) API failure, see reported status code or system log for details.
Definition at line 766 of file cfe_es_eventids.h.

12.112.2.13 CFE_ES_CREATING_CDS_DUMP_ERR_EID #define CFE_ES_CREATING_CDS_DUMP_ERR_EID 86
ES Write Critical Data Store Registry Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Write Critical Data Store Registry Command](#) failed to parse filename or open/create the file. OVERLOADED
Definition at line 953 of file cfe_es_eventids.h.

12.112.2.14 CFE_ES_ERLOG1_INF_EID #define CFE_ES_ERLOG1_INF_EID 19
ES Clear Exception Reset Log Command Success Event ID.

Type: INFORMATION

Cause:

[ES Clear Exception Reset Log Command](#) success.
Definition at line 239 of file cfe_es_eventids.h.

12.112.2.15 CFE_ES_ERLOG2_EID #define CFE_ES_ERLOG2_EID 20
ES Write Exception Reset Log Complete Event ID.

Type: DEBUG

Cause:

Request to write the Exception Reset log successfully completed.
Definition at line 250 of file cfe_es_eventids.h.

12.112.2.16 CFE_ES_ERLOG2_ERR_EID #define CFE_ES_ERLOG2_ERR_EID 56
ES Write Exception Reset Log Command Request or File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Write Exception Reset Log Command](#) request failed or file creation failed. OVERLOADED
Definition at line 626 of file cfe_es_eventids.h.

12.112.2.17 CFE_ES_ERLOG_PENDING_ERR_EID #define CFE_ES_ERLOG_PENDING_ERR_EID 93
ES Write Exception Reset Log Command Already In Progress Event ID.

Type: ERROR

Cause:

[ES Write Exception Reset Log Command](#) failure due to a write already being in progress.
Definition at line 1059 of file cfe_es_eventids.h.

12.112.2.18 CFE_ES_ERR_SYSLOGMODE_EID #define CFE_ES_ERR_SYSLOGMODE_EID 71
ES Set System Log Overwrite Mode Command Failed Event ID.

Type: ERROR

Cause:

[ES Set System Log Overwrite Mode Command](#) failed due to invalid mode requested.
Definition at line 789 of file cfe_es_eventids.h.

12.112.2.19 CFE_ES_ERREXIT_APP_ERR_EID #define CFE_ES_ERREXIT_APP_ERR_EID 33
ES Error Exit Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Error request to exit an application failed during application cleanup. Application and related resources will be in undefined state.
Definition at line 379 of file cfe_es_eventids.h.

12.112.2.20 CFE_ES_ERREXIT_APP_INF_EID #define CFE_ES_ERREXIT_APP_INF_EID 14
ES Error Exit Application Complete Event ID.

Type: INFORMATION

Cause:

Error request to exit an application successfully completed. This event indicates 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.

Definition at line 184 of file cfe_es_eventids.h.

12.112.2.21 CFE_ES_EXIT_APP_ERR_EID #define CFE_ES_EXIT_APP_ERR_EID 46
ES Exit Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Nominal request to exit an application failed during application cleanup. Application and related resources will be in undefined state.
Definition at line 522 of file cfe_es_eventids.h.

12.112.2.22 CFE_ES_EXIT_APP_INF_EID #define CFE_ES_EXIT_APP_INF_EID 13
ES Nominal Exit Application Complete Event ID.

Type: INFORMATION

Cause:

Nominal request to exit an application successfully completed. This event indicates the Application exited due to a nominal exit condition.

Definition at line 170 of file cfe_es_eventids.h.

12.112.2.23 CFE_ES_FILEWRITE_ERR_EID #define CFE_ES_FILEWRITE_ERR_EID 74
ES File Write Failed Event ID.

Type: ERROR

Cause:

ES File Write failure writing data to file. OVERLOADED

Definition at line 822 of file cfe_es_eventids.h.

12.112.2.24 CFE_ES_INIT_INF_EID #define CFE_ES_INIT_INF_EID 1
ES Initialization Event ID.

Type: INFORMATION

Cause:

Executive Services Task initialization complete.

Definition at line 42 of file cfe_es_eventids.h.

12.112.2.25 CFE_ES_INITSTATS_INF_EID #define CFE_ES_INITSTATS_INF_EID 2
ES Initialization Statistics Information Event ID.

Type: INFORMATION

Cause:

Executive Services Task initialization complete.

Definition at line 53 of file cfe_es_eventids.h.

12.112.2.26 CFE_ES_INVALID_POOL_HANDLE_ERR_EID #define CFE_ES_INVALID_POOL_HANDLE_ERR_EID 82
ES Telemeter Memory Statistics Command Invalid Handle Event ID.

Type: ERROR

Cause:

[ES Telemeter Memory Statistics Command](#) failure due to an invalid memory handle.
Definition at line 906 of file cfe_es_eventids.h.

12.112.2.27 CFE_ES_LEN_ERR_EID #define CFE_ES_LEN_ERR_EID 23
ES Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE_ES_CMD_MID](#) received on the ES message pipe.
Definition at line 283 of file cfe_es_eventids.h.

12.112.2.28 CFE_ES_MID_ERR_EID #define CFE_ES_MID_ERR_EID 21
ES Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the ES message pipe.
Definition at line 261 of file cfe_es_eventids.h.

12.112.2.29 CFE_ES_NOOP_INF_EID #define CFE_ES_NOOP_INF_EID 3
ES No-op Command Success Event ID.

Type: INFORMATION

Cause:

[ES No-op Command](#) success.
Definition at line 64 of file cfe_es_eventids.h.

12.112.2.30 CFE_ES_ONE_APP_EID #define CFE_ES_ONE_APP_EID 15
ES Query One Application Command Success Event ID.

Type: DEBUG

Cause:

[ES Query One Application Command](#) success.
Definition at line 195 of file cfe_es_eventids.h.

12.112.2.31 CFE_ES_ONE_APPID_ERR_EID #define CFE_ES_ONE_APPID_ERR_EID 50
ES Query One Application Data Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Query One Application Data Command](#) failed to get application ID from application name. Message will not be sent.
Definition at line 569 of file cfe_es_eventids.h.

12.112.2.32 CFE_ES_ONE_ERR_EID #define CFE_ES_ONE_ERR_EID 49
ES Query One Application Data Command Transmit Message Failed Event ID.

Type: ERROR

Cause:

[ES Query One Application Data Command](#) failed during message transmission.
Definition at line 557 of file cfe_es_eventids.h.

12.112.2.33 CFE_ES_OSCREATE_ERR_EID #define CFE_ES_OSCREATE_ERR_EID 51
ES Query All Application Data Command File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to create file.
Definition at line 580 of file cfe_es_eventids.h.

12.112.2.34 CFE_ES_PCR_ERR1_EID #define CFE_ES_PCR_ERR1_EID 47
ES Process Control Invalid Exception State Event ID.

Type: ERROR

Cause:

Invalid Exception state encountered when processing requests for application state changes. Exceptions are processed immediately, so this state should never occur during routine processing.

Definition at line 534 of file cfe_es_eventids.h.

12.112.2.35 CFE_ES_PCR_ERR2_EID #define CFE_ES_PCR_ERR2_EID 48
ES Process Control Unknown State Event ID.

Type: ERROR

Cause:

Unknown state encountered when processing requests for application state changes.

Definition at line 545 of file cfe_es_eventids.h.

12.112.2.36 CFE_ES_PERF_DATAWRITTEN_EID #define CFE_ES_PERF_DATAWRITTEN_EID 68
Performance Log Write Success Event ID.

Type: DEBUG

Cause:

Request to write the performance log successfully completed.

Definition at line 755 of file cfe_es_eventids.h.

12.112.2.37 CFE_ES_PERF_FILTMSKCMD_EID #define CFE_ES_PERF_FILTMSKCMD_EID 63
ES Set Performance Analyzer Filter Mask Command Success Event ID.

Type: DEBUG

Cause:

[ES Set Performance Analyzer Filter Mask Command](#) success.

Definition at line 697 of file cfe_es_eventids.h.

12.112.2.38 CFE_ES_PERF_FILTMSKERR_EID #define CFE_ES_PERF_FILTMSKERR_EID 64
ES Set Performance Analyzer Filter Mask Command Invalid Index Event ID.

Type: ERROR

Cause:

[ES Set Performance Analyzer Filter Mask Command](#) failed filter index range check.
Definition at line 709 of file cfe_es_eventids.h.

12.112.2.39 CFE_ES_PERF_LOG_ERR_EID #define CFE_ES_PERF_LOG_ERR_EID 67
ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) failed either parsing the file name or during open/creation of the file. OVERLOADED
Definition at line 744 of file cfe_es_eventids.h.

12.112.2.40 CFE_ES_PERF_STARTCMD_EID #define CFE_ES_PERF_STARTCMD_EID 57
ES Start Performance Analyzer Data Collection Command Success Event ID.

Type: DEBUG

Cause:

[ES Start Performance Analyzer Data Collection Command](#) success.
Definition at line 637 of file cfe_es_eventids.h.

12.112.2.41 CFE_ES_PERF_STARTCMD_ERR_EID #define CFE_ES_PERF_STARTCMD_ERR_EID 58
ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.

Type: ERROR

Cause:

[ES Start Performance Analyzer Data Collection Command](#) failed due to already being started.
Definition at line 649 of file cfe_es_eventids.h.

12.112.2.42 CFE_ES_PERF_STARTCMD_TRIG_ERR_EID #define CFE_ES_PERF_STARTCMD_TRIG_ERR_EID 59
ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.

Type: ERROR

Cause:

[ES Start Performance Analyzer Data Collection Command](#) failed due to invalid trigger mode.
Definition at line 661 of file cfe_es_eventids.h.

12.112.2.43 CFE_ES_PERF_STOPCMD_EID #define CFE_ES_PERF_STOPCMD_EID 60
ES Stop Performance Analyzer Data Collection Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) success. Note this event signifies the request to stop and write the performance data has been successfully submitted. The successful completion will generate a [CFE_ES_PERF_DATAWRITTEN_EID](#) event.
Definition at line 674 of file cfe_es_eventids.h.

12.112.2.44 CFE_ES_PERF_STOPCMD_ERR2_EID #define CFE_ES_PERF_STOPCMD_ERR2_EID 62
ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.

Type: ERROR

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) failed due to a write already in progress.
Definition at line 686 of file cfe_es_eventids.h.

12.112.2.45 CFE_ES_PERF_TRIGMSKCMD_EID #define CFE_ES_PERF_TRIGMSKCMD_EID 65
ES Set Performance Analyzer Trigger Mask Command Success Event ID.

Type: DEBUG

Cause:

[ES Set Performance Analyzer Trigger Mask Command](#) success.
Definition at line 720 of file cfe_es_eventids.h.

12.112.2.46 CFE_ES_PERF_TRIGMSKERR_EID #define CFE_ES_PERF_TRIGMSKERR_EID 66
ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.

Type: ERROR

Cause:

[ES Set Performance Analyzer Trigger Mask Command](#) failed the mask range check.
Definition at line 732 of file cfe_es_eventids.h.

12.112.2.47 CFE_ES_RELOAD_APP_DBG_EID #define CFE_ES_RELOAD_APP_DBG_EID 11
ES Reload Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Reload Application Command](#) success. Note this event signifies the request to reload the application has been successfully submitted. The successful completion will generate a [CFE_ES_RELOAD_APP_INF_EID](#) event.
Definition at line 147 of file cfe_es_eventids.h.

12.112.2.48 CFE_ES_RELOAD_APP_ERR1_EID #define CFE_ES_RELOAD_APP_ERR1_EID 42
ES Reload Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Reload Application Command](#) request failed.
Definition at line 473 of file cfe_es_eventids.h.

12.112.2.49 CFE_ES_RELOAD_APP_ERR2_EID #define CFE_ES_RELOAD_APP_ERR2_EID 43
ES Reload Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Reload Application Command](#) failed to get application ID from application name. The application will not be reloaded.
Definition at line 485 of file cfe_es_eventids.h.

12.112.2.50 CFE_ES_RELOAD_APP_ERR3_EID #define CFE_ES_RELOAD_APP_ERR3_EID 44
ES Reload Application Startup Failed Event ID.

Type: ERROR

Cause:

Request to reload an application failed during application startup. The application will not be reloaded.
Definition at line 497 of file cfe_es_eventids.h.

12.112.2.51 CFE_ES_RELOAD_APP_ERR4_EID #define CFE_ES_RELOAD_APP_ERR4_EID 45
ES Reload Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to reload an application failed during application cleanup. The application will not be reloaded and will be in an undefined state along with its associated resources.
Definition at line 510 of file cfe_es_eventids.h.

12.112.2.52 CFE_ES_RELOAD_APP_INF_EID #define CFE_ES_RELOAD_APP_INF_EID 12
ES Reload Application Complete Event ID.

Type: INFORMATION

Cause:

Request to reload an application successfully completed.
Definition at line 158 of file cfe_es_eventids.h.

12.112.2.53 CFE_ES_RESET_INF_EID #define CFE_ES_RESET_INF_EID 4
ES Reset Counters Command Success Event ID.

Type: INFORMATION

Cause:

[ES Reset Counters Command](#) success.
Definition at line 75 of file cfe_es_eventids.h.

12.112.2.54 CFE_ES_RESET_PR_COUNT_EID #define CFE_ES_RESET_PR_COUNT_EID 72
ES Set Processor Reset Counter to Zero Command Success Event ID.

Type: INFORMATION

Cause:

[ES Set Processor Reset Counter to Zero Command](#) success.

Definition at line 800 of file cfe_es_eventids.h.

12.112.2.55 CFE_ES_RESTART_APP_DBG_EID #define CFE_ES_RESTART_APP_DBG_EID 9
ES Restart Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Restart Application Command](#) success. Note this event signifies the request to restart the application has been successfully submitted. The successful completion will generate a [CFE_ES_RESTART_APP_INF_EID](#) event.

Definition at line 123 of file cfe_es_eventids.h.

12.112.2.56 CFE_ES_RESTART_APP_ERR1_EID #define CFE_ES_RESTART_APP_ERR1_EID 38
ES Restart Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Restart Application Command](#) request failed.

Definition at line 425 of file cfe_es_eventids.h.

12.112.2.57 CFE_ES_RESTART_APP_ERR2_EID #define CFE_ES_RESTART_APP_ERR2_EID 39
ES Restart Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Restart Application Command](#) failed to get application ID from application name. The application will not be restarted.

Definition at line 437 of file cfe_es_eventids.h.

12.112.2.58 CFE_ES_RESTART_APP_ERR3_EID #define CFE_ES_RESTART_APP_ERR3_EID 40
ES Restart Application Startup Failed Event ID.

Type: ERROR

Cause:

Request to restart an application failed during application startup. The application will not be restarted.
Definition at line 449 of file cfe_es_eventids.h.

12.112.2.59 CFE_ES_RESTART_APP_ERR4_EID #define CFE_ES_RESTART_APP_ERR4_EID 41
ES Restart Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to restart an application failed during application cleanup. The application will not be restarted and will be in an undefined state along with its associated resources.
Definition at line 462 of file cfe_es_eventids.h.

12.112.2.60 CFE_ES_RESTART_APP_INF_EID #define CFE_ES_RESTART_APP_INF_EID 10
ES Restart Application Completed Event ID.

Type: INFORMATION

Cause:

Request to restart an application successfully completed.
Definition at line 134 of file cfe_es_eventids.h.

12.112.2.61 CFE_ES_SET_MAX_PR_COUNT_EID #define CFE_ES_SET_MAX_PR_COUNT_EID 73
ES Set Maximum Processor Reset Limit Command Success Event ID.

Type: INFORMATION

Cause:

[ES Set Maximum Processor Reset Limit Command](#) success.
Definition at line 811 of file cfe_es_eventids.h.

12.112.2.62 CFE_ES_START_ERR_EID #define CFE_ES_START_ERR_EID 26
ES Start Application Command Application Creation Failed Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure during application creation after successful parameter validation.
Definition at line 306 of file cfe_es_eventids.h.

12.112.2.63 CFE_ES_START_EXC_ACTION_ERR_EID #define CFE_ES_START_EXC_ACTION_ERR_EID 32
ES Start Application Command Exception Action Invalid Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to invalid application exception action.
Definition at line 367 of file cfe_es_eventids.h.

12.112.2.64 CFE_ES_START_INF_EID #define CFE_ES_START_INF_EID 6
ES Start Application Command Success Event ID.

Type: INFORMATION

Cause:

[ES Start Application Command](#) success.
Definition at line 86 of file cfe_es_eventids.h.

12.112.2.65 CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID #define CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID 28
ES Start Application Command Entry Point NULL Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to a NULL Application Entry Point.
Definition at line 330 of file cfe_es_eventids.h.

12.112.2.66 CFE_ES_START_INVALID_FILENAME_ERR_EID #define CFE_ES_START_INVALID_FILENAME_ERR_EID 27

ES Start Application Command Invalid Filename Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to invalid filename.

Definition at line 318 of file cfe_es_eventids.h.

12.112.2.67 CFE_ES_START_NULL_APP_NAME_ERR_EID #define CFE_ES_START_NULL_APP_NAME_ERR_EID 29

ES Start Application Command App Name NULL Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to NULL Application Name.

Definition at line 342 of file cfe_es_eventids.h.

12.112.2.68 CFE_ES_START_PRIORITY_ERR_EID #define CFE_ES_START_PRIORITY_ERR_EID 31

ES Start Application Command Priority Too Large Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to a requested application priority greater than the maximum priority allowed for tasks as defined by the OS Abstraction Layer (OS_MAX_PRIORITY).

Definition at line 355 of file cfe_es_eventids.h.

12.112.2.69 CFE_ES_STOP_DBG_EID #define CFE_ES_STOP_DBG_EID 7

ES Stop Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Stop Application Command](#) success. Note this event signifies the request to delete the application has been successfully submitted. The successful completion will generate a [CFE_ES_STOP_INF_EID](#) event.

Definition at line 99 of file cfe_es_eventids.h.

12.112.2.70 CFE_ES_STOP_ERR1_EID #define CFE_ES_STOP_ERR1_EID 35
ES Stop Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Stop Application Command](#) request failed.
Definition at line 390 of file cfe_es_eventids.h.

12.112.2.71 CFE_ES_STOP_ERR2_EID #define CFE_ES_STOP_ERR2_EID 36
ES Stop Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Stop Application Command](#) failed to get application ID from application name. The application will not be deleted.
Definition at line 402 of file cfe_es_eventids.h.

12.112.2.72 CFE_ES_STOP_ERR3_EID #define CFE_ES_STOP_ERR3_EID 37
ES Stop Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to delete an application failed during application cleanup. Application and related resources will be in undefined state.
Definition at line 414 of file cfe_es_eventids.h.

12.112.2.73 CFE_ES_STOP_INF_EID #define CFE_ES_STOP_INF_EID 8
ES Stop Application Completed Event ID.

Type: INFORMATION

Cause:

Request to delete an application successfully completed.
Definition at line 110 of file cfe_es_eventids.h.

12.112.2.74 CFE_ES_SYSLOG1_INF_EID #define CFE_ES_SYSLOG1_INF_EID 17
ES Clear System Log Command Success Event ID.

Type: INFORMATION

Cause:

[ES Clear System Log Command](#) success.
Definition at line 217 of file cfe_es_eventids.h.

12.112.2.75 CFE_ES_SYSLOG2_EID #define CFE_ES_SYSLOG2_EID 18
ES Write System Log Command Success Event ID.

Type: DEBUG

Cause:

[ES Write System Log Command](#) success.
Definition at line 228 of file cfe_es_eventids.h.

12.112.2.76 CFE_ES_SYSLOG2_ERR_EID #define CFE_ES_SYSLOG2_ERR_EID 55
ES Write System Log Command Filename Parse or File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Write System Log Command](#) failed parsing file name or creating the file. OVERLOADED
Definition at line 614 of file cfe_es_eventids.h.

12.112.2.77 CFE_ES_SYSLOGMODE_EID #define CFE_ES_SYSLOGMODE_EID 70
ES Set System Log Overwrite Mode Command Success Event ID.

Type: DEBUG

Cause:

[ES Set System Log Overwrite Mode Command](#) success.
Definition at line 777 of file cfe_es_eventids.h.

12.112.2.78 CFE_ES_TASKINFO_EID #define CFE_ES_TASKINFO_EID 87
ES Write All Task Data Command Success Event ID.

Type: DEBUG

Cause:

[ES Write All Task Data Command](#) success.
Definition at line 964 of file cfe_es_eventids.h.

12.112.2.79 CFE_ES_TASKINFO_OSCREATE_ERR_EID #define CFE_ES_TASKINFO_OSCREATE_ERR_EID 88
ES Write All Task Data Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to parse the filename or open/create the file.
Definition at line 976 of file cfe_es_eventids.h.

12.112.2.80 CFE_ES_TASKINFO_WR_ERR_EID #define CFE_ES_TASKINFO_WR_ERR_EID 90
ES Write All Task Data Command Write Data Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to write task data to file.
Definition at line 1000 of file cfe_es_eventids.h.

12.112.2.81 CFE_ES_TASKINFO_WRHDR_ERR_EID #define CFE_ES_TASKINFO_WRHDR_ERR_EID 89
ES Write All Task Data Command Write Header Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to write file header.
Definition at line 988 of file cfe_es_eventids.h.

12.112.2.82 CFE_ES_TASKWR_ERR_EID #define CFE_ES_TASKWR_ERR_EID 53
ES Query All Application Data Command File Write App Data Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to write file application data.
Definition at line 602 of file cfe_es_eventids.h.

12.112.2.83 CFE_ES_TLM_POOL_STATS_INFO_EID #define CFE_ES_TLM_POOL_STATS_INFO_EID 81
ES Telemeter Memory Statistics Command Success Event ID.

Type: DEBUG

Cause:

[ES Telemeter Memory Statistics Command](#) success.
Definition at line 894 of file cfe_es_eventids.h.

12.112.2.84 CFE_ES_VERSION_INF_EID #define CFE_ES_VERSION_INF_EID 91
cFS Version Information Event ID

Type: INFORMATION

Cause:

ES Initialization complete and response to [ES NO-OP Command](#).
A separate version info event will be generated for every module which is statically linked into the CFE core executable (e.g. OSAL, PSP, MSG, SBR, etc).
The version information reported in this event is derived from the source revision control system at build time, as opposed to manually-assigned semantic version numbers. It is intended to uniquely identify the actual source code that is currently running, to the extent this is possible.
The Mission version information also identifies the build configuration name, if available.
Definition at line 1021 of file cfe_es_eventids.h.

12.112.2.85 CFE_ES_WRHDR_ERR_EID #define CFE_ES_WRHDR_ERR_EID 52
ES Query All Application Data Command File Write Header Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to write file header.
Definition at line 591 of file cfe_es_eventids.h.

12.112.2.86 CFE_ES_WRITE_CFE_HDR_ERR_EID #define CFE_ES_WRITE_CFE_HDR_ERR_EID 85
ES Write Critical Data Store Registry Command Header Write Failed Event ID.

Type: ERROR

Cause:

ES Write Critical Data Store Registry Command failed to write header.
Definition at line 941 of file cfe_es_eventids.h.

12.113 cfe/modules/evs/config/default_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.

12.113.1 Detailed Description

Declarations and prototypes for cfe_evs_extern_typedefs module

12.113.2 Typedef Documentation

12.113.2.1 CFE_EVS_EventFilter_Enum_t `typedef uint8 CFE_EVS_EventFilter_Enum_t`
Identifies event filter schemes.

See also

enum [CFE_EVS_EventFilter](#)

Definition at line 125 of file default_cfe_evs_extern_typedefs.h.

12.113.2.2 CFE_EVS_EventOutput_Enum_t `typedef uint8 CFE_EVS_EventOutput_Enum_t`
Identifies event output port.

See also

enum [CFE_EVS_EventOutput](#)

Definition at line 158 of file default_cfe_evs_extern_typedefs.h.

12.113.2.3 CFE_EVS_EventType_Enum_t `typedef uint16 CFE_EVS_EventType_Enum_t`
Identifies type of event message.

See also

enum [CFE_EVS_EventType](#)

Definition at line 107 of file default_cfe_evs_extern_typedefs.h.

12.113.2.4 CFE_EVS_LogMode_Enum_t `typedef uint8 CFE_EVS_LogMode_Enum_t`
Identifies handling of log messages after storage is filled.

See also

enum [CFE_EVS_LogMode](#)

Definition at line 74 of file default_cfe_evs_extern_typedefs.h.

12.113.2.5 CFE_EVS_MsgFormat_Enum_t `typedef uint8 CFE_EVS_MsgFormat_Enum_t`
Identifies format of log messages.

See also

enum [CFE_EVS_MsgFormat](#)

Definition at line 51 of file default_cfe_evs_extern_typedefs.h.

12.113.3 Enumeration Type Documentation

12.113.3.1 CFE_EVS_EventFilter `enum CFE_EVS_EventFilter`
Label definitions associated with CFE_EVS_EventFilter_Enum_t.

Enumerator

<code>CFE_EVS_EventFilter_BINARY</code>	Binary event filter.
---	----------------------

Definition at line 112 of file default_cfe_evs_extern_typedefs.h.

12.113.3.2 CFE_EVS_EventOutput enum [CFE_EVS_EventOutput](#)

Label definitions associated with CFE_EVS_EventOutput_Enum_t.

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 130 of file default_cfe_evs_extern_typedefs.h.

12.113.3.3 CFE_EVS_EventType enum [CFE_EVS_EventType](#)

Label definitions associated with CFE_EVS_EventType_Enum_t.

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 79 of file default_cfe_evs_extern_typedefs.h.

12.113.3.4 CFE_EVS_LogMode enum [CFE_EVS_LogMode](#)

Label definitions associated with CFE_EVS_LogMode_Enum_t.

Enumerator

CFE_EVS_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_EVS_LogMode_DISCARD	Discard Log Mode.

Definition at line 56 of file default_cfe_evs_extern_typedefs.h.

12.113.3.5 CFE_EVS_MsgFormat enum [CFE_EVS_MsgFormat](#)

Label definitions associated with CFE_EVS_MsgFormat_Enum_t.

Enumerator

CFE_EVS_MsgFormat_SHORT	Short Format Messages.
CFE_EVS_MsgFormat_LONG	Long Format Messages.

Definition at line 33 of file default_cfe_evs_extern_typedefs.h.

12.114 cfe/modules/evs/config/default_cfe_evs_fcncodes.h File Reference

Macros

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

12.114.1 Detailed Description

Specification for the CFE Event Services (CFE_EVS) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.114.2 Macro Definition Documentation

12.114.2.1 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_AddEventFilterCmd_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):

- Application name is not valid or not registered with event services
- Specified event ID is already added to the application event filter
- Maximum number of event IDs already added to filter

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 693 of file default_cfe_evs_fcncodes.h.

12.114.2.2 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_EVS_ClearLogCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_EVS_CMDPC** - command execution counter will increment
- **\$sc_\$cpu_EVS_LOGFULL** - The LogFullFlag in the Housekeeping telemetry will be cleared
- **\$sc_\$cpu_EVS_LOGOVERFLOWC** - The LogOverflowCounter in the Housekeeping telemetry will be reset to 0

Error Conditions

There are no error conditions for this command. If the Event Services receives the command, the log is cleared.

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 873 of file default_cfe_evs_fcncodes.h.

12.114.2.3 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_DeleteEventFilterCmd_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):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

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 728 of file default_cfe_evs_fcncodes.h.

12.114.2.4 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_DisableAppEventTypeCmd_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):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set
- Application name is not valid or not registered with event services

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_APP_EVENT_TYPE_CC](#),
[CFE_EVS_ENABLE_APP_EVENTS_CC](#), [CFE_EVS_DISABLE_APP_EVENTS_CC](#)

Definition at line 353 of file default_cfe_evs_fcncodes.h.

12.114.2.5 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_DisableAppEventsCmd_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):

- Application name is not valid or not registered with event services

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_APP_EVENT_TYPE_CC`,
`CFE_EVS_DISABLE_APP_EVENT_TYPE_CC`, `CFE_EVS_ENABLE_APP_EVENTS_CC`

Definition at line 431 of file default_cfe_evs_fcncodes.h.

12.114.2.6 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_DisableEventTypeCmd_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):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

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 201 of file default_cfe_evs_fcncodes.h.

12.114.2.7 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_DisablePortsCmd_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):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

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 587 of file default_cfe_evs_fcncodes.h.

12.114.2.8 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_EnableAppEventTypeCmd_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_ENAAPPLEVTTYPE_EID` debug event message

Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set
- Application name is not valid or not registered with event services

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_APP_EVENT_TYPE_CC](#),
[CFE_EVS_ENABLE_APP_EVENTS_CC](#), [CFE_EVS_DISABLE_APP_EVENTS_CC](#)

Definition at line 300 of file default_cfe_evs_fcncodes.h.

12.114.2.9 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_EnableAppEventsCmd_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_ENAAPPLEVT_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):

- Application name is not valid or not registered with event services

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_APP_EVENT_TYPE_CC](#),
[CFE_EVS_DISABLE_APP_EVENT_TYPE_CC](#), [CFE_EVS_DISABLE_APP_EVENTS_CC](#)

Definition at line 392 of file default_cfe_evs_fcncodes.h.

12.114.2.10 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_EnableEventTypeCmd_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):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

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_DISABLE_APP_EVENT_TYPE_CC](#),
[CFE_EVS_ENABLE_APP_EVENTS_CC](#), [CFE_EVS_DISABLE_APP_EVENTS_CC](#)

Definition at line 152 of file default_cfe_evs_fcncodes.h.

12.114.2.11 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_EnablePortsCmd_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):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

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 548 of file default_cfe_evs_fcncodes.h.

12.114.2.12 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_EVS_NoopCmd_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 65 of file default_cfe_evs_fcncodes.h.

12.114.2.13 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_ResetAllFiltersCmd_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):

- Application name is not valid or not registered with event services

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 657 of file default_cfe_evs_fncodes.h.

12.114.2.14 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_ResetAppCounterCmd_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):

- Application name is not valid or not registered with event services

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 467 of file default_cfe_evs_fcncodes.h.

12.114.2.15 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_EVS_ResetCountersCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_EVS_CMDPC** - command execution counter will be reset to 0
- **\$sc_\$cpu_EVS_CMDEC** - command error counter will be reset to 0
- 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 104 of file default_cfe_evs_fcncodes.h.

12.114.2.16 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_ResetFilterCmd_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):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

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 623 of file default_cfe_evs_fcncodes.h.

12.114.2.17 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_SetEventFormatModeCmd_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 MsgFormat 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 248 of file default_cfe_evs_fcncodes.h.

12.114.2.18 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_SetFilterCmd_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):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

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 509 of file default_cfe_evs_fcncodes.h.

12.114.2.19 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_SetLogModeCmd_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 LogMode selected - must be either [CFE_EVS_LogMode_OVERWRITE](#) or [CFE_EVS_LogMode_DISCARD](#)

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 838 of file default_cfe_evs_fcncodes.h.

12.114.2.20 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_WriteAppDataFileCmd_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 file specified in the command (or the default specified by the [CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE](#) configuration parameter) will be updated with the latest information.

Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

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 767 of file default_cfe_evs_fcncodes.h.

12.114.2.21 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_WriteLogFileCmd_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):

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

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 802 of file default_cfe_evs_fcncodes.h.

12.115 cfe/modules/evs/config/default_cfe_evs_interface_cfg.h File Reference

Macros

- `#define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122`

12.115.1 Detailed Description

CFE Event Services (CFE_EVS) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.115.2 Macro Definition Documentation

12.115.2.1 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

This length does not need to include an extra character for NULL termination.

Limits

Not Applicable

Definition at line 47 of file default_cfe_evs_interface_cfg.h.

12.116 cfe/modules/evs/config/default_cfe_evs_internal_cfg.h File Reference

Macros

- #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_EVS_MAX_EVENT_FILTERS 8
- #define CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST 32
- #define CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC 15
- #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

12.116.1 Detailed Description

CFE Event Services (CFE_EVS) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.116.2 Macro Definition Documentation

12.116.2.1 CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC #define CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC 15

Purpose Sustained number of event messages per second per app before squelching

Description:

Sustained number of events that may be emitted per app per second.

Limits

This number must be less than or equal to CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST. Values lower than 8 may cause functional and unit test failures.

Definition at line 96 of file default_cfe_evs_internal_cfg.h.

12.116.2.2 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 137 of file default_cfe_evs_internal_cfg.h.

12.116.2.3 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 110 of file default_cfe_evs_internal_cfg.h.

12.116.2.4 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 184 of file default_cfe_evs_internal_cfg.h.

12.116.2.5 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 197 of file default_cfe_evs_internal_cfg.h.

12.116.2.6 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG #define CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG←
AG 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 168 of file default_cfe_evs_internal_cfg.h.

12.116.2.7 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 122 of file default_cfe_evs_internal_cfg.h.

12.116.2.8 CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST #define CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST 32

Purpose Maximum number of event before squelching

Description:

Maximum number of events that may be emitted per app per second. Setting this to 0 will cause events to be unrestricted.

Limits

This number must be less than or equal to INT_MAX/1000

Definition at line 84 of file default_cfe_evs_internal_cfg.h.

12.116.2.9 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 72 of file default_cfe_evs_internal_cfg.h.

12.116.2.10 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 disabled) 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 151 of file default_cfe_evs_internal_cfg.h.

12.116.2.11 CFE_PLATFORM_EVS_START_TASK_PRIORITY #define CFE_PLATFORM_EVS_START_TASK_PRIO←
RITY 61

Purpose Define EVS Task Priority

Description:

Defines the cFE_EVS Task priority.

Limits

Not Applicable

Definition at line 44 of file default_cfe_evs_internal_cfg.h.

12.116.2.12 CFE_PLATFORM_EVS_START_TASK_STACK_SIZE #define CFE_PLATFORM_EVS_START_TASK_S←
TACK_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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 59 of file default_cfe_evs_internal_cfg.h.

12.117 cfe/modules/evs/config/default_cfe_evs_mission_cfg.h File Reference

```
#include "cfe_evs_interface_cfg.h"
```

12.117.1 Detailed Description

CFE Event Services (CFE_EVS) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.118 cfe/modules/evs/config/default_cfe_evs_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_evs_fcncodes.h"
#include "cfe_evs_msgdefs.h"
#include "cfe_evs_msgstruct.h"
```

12.118.1 Detailed Description

Specification for the CFE Event Services (CFE_EVS) command and telemetry message data types.

This is a compatibility header for the "cfe_evs_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.119 cfe/modules/evs/config/default_cfe_evs_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_evs_fcncodes.h"
```

Data Structures

- struct [CFE_EVS_LogFileCmd_Payload](#)
Write Event Log to File Command Payload.
- struct [CFE_EVS_AppDataCmd_Payload](#)
Write Event Services Application Information to File Command Payload.
- struct [CFE_EVS_SetLogMode_Payload](#)
Set Log Mode Command Payload.
- struct [CFE_EVS_SetEventFormatCode_Payload](#)
Set Event Format Mode Command Payload.
- struct [CFE_EVS_BitMaskCmd_Payload](#)
Generic Bitmask Command Payload.

- struct [CFE_EVS_AppNameCmd_Payload](#)
Generic App Name Command Payload.
- struct [CFE_EVS_AppNameEventIDCmd_Payload](#)
Generic App Name and Event ID Command Payload.
- struct [CFE_EVS_AppNameBitMaskCmd_Payload](#)
Generic App Name and Bitmask Command Payload.
- struct [CFE_EVS_AppNameEventIDMaskCmd_Payload](#)
Generic App Name, Event ID, Mask Command Payload.
- struct [CFE_EVS_AppTlmData](#)
- struct [CFE_EVS_HousekeepingTlm_Payload](#)
- struct [CFE_EVS_PacketID](#)
- struct [CFE_EVS_LongEventTlm_Payload](#)
- struct [CFE_EVS_ShortEventTlm_Payload](#)

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

Typedefs

- typedef struct [CFE_EVS_LogFileCmd_Payload](#) [CFE_EVS_LogFileCmd_Payload_t](#)
Write Event Log to File Command Payload.
- typedef struct [CFE_EVS_AppDataCmd_Payload](#) [CFE_EVS_AppDataCmd_Payload_t](#)
Write Event Services Application Information to File Command Payload.
- typedef struct [CFE_EVS_SetLogMode_Payload](#) [CFE_EVS_SetLogMode_Payload_t](#)
Set Log Mode Command Payload.
- typedef struct [CFE_EVS_SetEventFormatCode_Payload](#) [CFE_EVS_SetEventFormatMode_Payload_t](#)
Set Event Format Mode Command Payload.
- typedef struct [CFE_EVS_BitMaskCmd_Payload](#) [CFE_EVS_BitMaskCmd_Payload_t](#)
Generic Bitmask Command Payload.
- typedef struct [CFE_EVS_AppNameCmd_Payload](#) [CFE_EVS_AppNameCmd_Payload_t](#)
Generic App Name Command Payload.
- typedef struct [CFE_EVS_AppNameEventIDCmd_Payload](#) [CFE_EVS_AppNameEventIDCmd_Payload_t](#)
Generic App Name and Event ID Command Payload.
- typedef struct [CFE_EVS_AppNameBitMaskCmd_Payload](#) [CFE_EVS_AppNameBitMaskCmd_Payload_t](#)
Generic App Name and Bitmask Command Payload.
- typedef struct [CFE_EVS_AppNameEventIDMaskCmd_Payload](#) [CFE_EVS_AppNameEventIDMaskCmd_Payload_t](#)
Generic App Name, Event ID, Mask Command Payload.
- typedef struct [CFE_EVS_AppTlmData](#) [CFE_EVS_AppTlmData_t](#)
- typedef struct [CFE_EVS_HousekeepingTlm_Payload](#) [CFE_EVS_HousekeepingTlm_Payload_t](#)
- typedef struct [CFE_EVS_PacketID](#) [CFE_EVS_PacketID_t](#)
- typedef struct [CFE_EVS_LongEventTlm_Payload](#) [CFE_EVS_LongEventTlm_Payload_t](#)
- typedef struct [CFE_EVS_ShortEventTlm_Payload](#) [CFE_EVS_ShortEventTlm_Payload_t](#)

12.119.1 Detailed Description

Specification for the CFE Event Services (CFE_EVS) command and telemetry message constant definitions.
For CFE_EVS this is only the function/command code definitions

12.119.2 Macro Definition Documentation

12.119.2.1 CFE_EVS_CRITICAL_BIT #define CFE_EVS_CRITICAL_BIT 0x0008

Definition at line 39 of file default_cfe_evs_msgdefs.h.

12.119.2.2 CFE_EVS_DEBUG_BIT #define CFE_EVS_DEBUG_BIT 0x0001

Definition at line 36 of file default_cfe_evs_msgdefs.h.

12.119.2.3 CFE_EVS_ERROR_BIT #define CFE_EVS_ERROR_BIT 0x0004

Definition at line 38 of file default_cfe_evs_msgdefs.h.

12.119.2.4 CFE_EVS_INFORMATION_BIT #define CFE_EVS_INFORMATION_BIT 0x0002

Definition at line 37 of file default_cfe_evs_msgdefs.h.

12.119.2.5 CFE_EVS_PORT1_BIT #define CFE_EVS_PORT1_BIT 0x0001

Definition at line 42 of file default_cfe_evs_msgdefs.h.

12.119.2.6 CFE_EVS_PORT2_BIT #define CFE_EVS_PORT2_BIT 0x0002

Definition at line 43 of file default_cfe_evs_msgdefs.h.

12.119.2.7 CFE_EVS_PORT3_BIT #define CFE_EVS_PORT3_BIT 0x0004

Definition at line 44 of file default_cfe_evs_msgdefs.h.

12.119.2.8 CFE_EVS_PORT4_BIT #define CFE_EVS_PORT4_BIT 0x0008

Definition at line 45 of file default_cfe_evs_msgdefs.h.

12.119.3 Typedef Documentation

12.119.3.1 CFE_EVS_AppDataCmd_Payload_t typedef struct CFE_EVS_AppDataCmd_Payload CFE_EVS_AppDataCmd_Payload_t

Write Event Services Application Information to File Command Payload.

For command details, see [CFE_EVS_WRITE_APP_DATA_FILE_CC](#)

12.119.3.2 CFE_EVS_AppNameBitMaskCmd_Payload_t typedef struct CFE_EVS_AppNameBitMaskCmd_Payload CFE_EVS_AppNameBitMaskCmd_Payload_t

Generic App Name and Bitmask Command Payload.

For command details, see [CFE_EVS_ENABLE_APP_EVENT_TYPE_CC](#) and/or [CFE_EVS_DISABLE_APP_EVENT_TYPE_CC](#)

12.119.3.3 CFE_EVS_AppNameCmd_Payload_t `typedef struct CFE_EVS_AppNameCmd_Payload CFE_EVS_AppNameCmd_Payload_t`
Generic App Name Command Payload.

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](#)

12.119.3.4 CFE_EVS_AppNameEventIDCmd_Payload_t `typedef struct CFE_EVS_AppNameEventIDCmd_Payload CFE_EVS_AppNameEventIDCmd_Payload_t`

Generic App Name and Event ID Command Payload.

For command details, see [CFE_EVS_RESET_FILTER_CC](#) and [CFE_EVS_DELETE_EVENT_FILTER_CC](#)

12.119.3.5 CFE_EVS_AppNameEventIDMaskCmd_Payload_t `typedef struct CFE_EVS_AppNameEventIDMaskCmd_Payload CFE_EVS_AppNameEventIDMaskCmd_Payload_t`

Generic App Name, Event ID, Mask Command Payload.

For command details, see [CFE_EVS_SET_FILTER_CC](#), [CFE_EVS_ADD_EVENT_FILTER_CC](#) and/or [CFE_EVS_DELETE_EVENT_FILTER_CC](#)

12.119.3.6 CFE_EVS_AppTlmData_t `typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t`

12.119.3.7 CFE_EVS_BitMaskCmd_Payload_t `typedef struct CFE_EVS_BitMaskCmd_Payload CFE_EVS_BitMaskCmd_Payload_t`
Generic Bitmask Command Payload.

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](#)

12.119.3.8 CFE_EVS_HousekeepingTlm_Payload_t `typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t`

Name Event Services Housekeeping Telemetry Packet

12.119.3.9 CFE_EVS_LogFileCmd_Payload_t `typedef struct CFE_EVS_LogFileCmd_Payload CFE_EVS_LogFileCmd_Payload_t`
Write Event Log to File Command Payload.

For command details, see [CFE_EVS_WRITE_LOG_DATA_FILE_CC](#)

12.119.3.10 CFE_EVS_LongEventTlm_Payload_t `typedef struct CFE_EVS_LongEventTlm_Payload CFE_EVS_LongEventTlm_Payload_t`

Name Event Message Telemetry Packet (Long format)

12.119.3.11 CFE_EVS_PacketID_t `typedef struct CFE_EVS_PacketID CFE_EVS_PacketID_t`
Telemetry packet structures

12.119.3.12 CFE_EVS_SetEventFormatMode_Payload_t `typedef struct CFE_EVS_SetEventFormatCode_Payload CFE_EVS_SetEventFormatMode_Payload_t`

Set Event Format Mode Command Payload.

For command details, see [CFE_EVS_SET_EVENT_FORMAT_MODE_CC](#)

12.119.3.13 CFE_EVS_SetLogMode_Payload_t `typedef struct CFE_EVS_SetLogMode_Payload CFE_EVS_SetLogMode_Payload_t`
Set Log Mode Command Payload.

For command details, see [CFE_EVS_SET_LOG_MODE_CC](#)

12.119.3.14 CFE_EVS_ShortEventTlm_Payload_t `typedef struct CFE_EVS_ShortEventTlm_Payload CFE_EVS_ShortEventTlm_`

Name Event Message Telemetry Packet (Short format)

12.120 cfe/modules/evs/config/default_cfe_evs_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_evs_topicids.h"
```

Macros

- `#define CFE_EVS_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_EVS_CMD_TOPICID)
/* 0x1801 */`
- `#define CFE_EVS_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_EVS_SEND_HK_TOPICID)
/* 0x1809 */`
- `#define CFE_EVS_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_HK_TLM_TOPICID)
/* 0x0801 */`
- `#define CFE_EVS_LONG_EVENT_MSG_MID`
- `#define CFE_EVS_SHORT_EVENT_MSG_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_SHORT_EVENT_MSG_MID)
/* 0x0809 */`

12.120.1 Detailed Description

CFE Event Services (CFE_EVS) Application Message IDs

12.120.2 Macro Definition Documentation**12.120.2.1 CFE_EVS_CMD_MID** `#define CFE_EVS_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_EVS_CMD_TOPICID)
/* 0x1801 */`

Definition at line 32 of file default_cfe_evs_msgids.h.

12.120.2.2 CFE_EVS_HK_TLM_MID `#define CFE_EVS_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_HK_TLM_TOPICID)
/* 0x0801 */`

Definition at line 38 of file default_cfe_evs_msgids.h.

12.120.2.3 CFE_EVS_LONG_EVENT_MSG_MID `#define CFE_EVS_LONG_EVENT_MSG_MID
CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID) /* 0x0808 */`

Definition at line 39 of file default_cfe_evs_msgids.h.

12.120.2.4 CFE_EVS_SEND_HK_MID `#define CFE_EVS_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_EVS_SEND_HK_TOPICID)
/* 0x1809 */`

Definition at line 33 of file default_cfe_evs_msgids.h.

12.120.2.5 CFE_EVS_SHORT_EVENT_MSG_MID `#define CFE_EVS_SHORT_EVENT_MSG_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID)
/* 0x0809 */`

Definition at line 41 of file default_cfe_evs_msgids.h.

12.121 cfe/modules/evs/config/default_cfe_evs_msgstruct.h File Reference

```
#include "common_types.h"
#include "cfe_evs_msgdefs.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- struct [CFE_EVS_NoopCmd](#)
- struct [CFE_EVS_ResetCountersCmd](#)
- struct [CFE_EVS_ClearLogCmd](#)
- struct [CFE_EVS_SendHkCmd](#)
- struct [CFE_EVS_WriteLogFileCmd](#)
Write Event Log to File Command.
- struct [CFE_EVS_WriteAppDataFileCmd](#)
Write Event Services Application Information to File Command.
- struct [CFE_EVS_SetLogModeCmd](#)
Set Log Mode Command.
- struct [CFE_EVS_SetEventFormatModeCmd](#)
Set Event Format Mode Command.
- struct [CFE_EVS_EnablePortsCmd](#)
- struct [CFE_EVS_DisablePortsCmd](#)
- struct [CFE_EVS_EnableEventTypeCmd](#)
- struct [CFE_EVS_DisableEventTypeCmd](#)
- struct [CFE_EVS_EnableAppEventsCmd](#)
- struct [CFE_EVS_DisableAppEventsCmd](#)
- struct [CFE_EVS_ResetAppCounterCmd](#)
- struct [CFE_EVS_ResetAllFiltersCmd](#)
- struct [CFE_EVS_ResetFilterCmd](#)
- struct [CFE_EVS_DeleteEventFilterCmd](#)
- struct [CFE_EVS_EnableAppEventTypeCmd](#)
- struct [CFE_EVS_DisableAppEventTypeCmd](#)
- struct [CFE_EVS_AddEventFilterCmd](#)
- struct [CFE_EVS_SetFilterCmd](#)
- struct [CFE_EVS_HousekeepingTlm](#)
- struct [CFE_EVS_LongEventTlm](#)
- struct [CFE_EVS_ShortEventTlm](#)

TypeDefs

- typedef struct [CFE_EVS_NoopCmd](#) CFE_EVS_NoopCmd_t
- typedef struct [CFE_EVS_ResetCountersCmd](#) CFE_EVS_ResetCountersCmd_t
- typedef struct [CFE_EVS_ClearLogCmd](#) CFE_EVS_ClearLogCmd_t
- typedef struct [CFE_EVS_SendHkCmd](#) CFE_EVS_SendHkCmd_t
- typedef struct [CFE_EVS_WriteLogFileCmd](#) CFE_EVS_WriteLogFileCmd_t
Write Event Log to File Command.
- typedef struct [CFE_EVS_WriteAppDataFileCmd](#) CFE_EVS_WriteAppDataFileCmd_t
Write Event Services Application Information to File Command.
- typedef struct [CFE_EVS_SetLogModeCmd](#) CFE_EVS_SetLogModeCmd_t

Set Log Mode Command.

- `typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t`
Set Event Format Mode Command.
- `typedef struct CFE_EVS_EnablePortsCmd CFE_EVS_EnablePortsCmd_t`
- `typedef struct CFE_EVS_DisablePortsCmd CFE_EVS_DisablePortsCmd_t`
- `typedef struct CFE_EVS_EnableEventTypeCmd CFE_EVS_EnableEventTypeCmd_t`
- `typedef struct CFE_EVS_DisableEventTypeCmd CFE_EVS_DisableEventTypeCmd_t`
- `typedef struct CFE_EVS_EnableAppEventsCmd CFE_EVS_EnableAppEventsCmd_t`
- `typedef struct CFE_EVS_DisableAppEventsCmd CFE_EVS_DisableAppEventsCmd_t`
- `typedef struct CFE_EVS_ResetAppCounterCmd CFE_EVS_ResetAppCounterCmd_t`
- `typedef struct CFE_EVS_ResetAllFiltersCmd CFE_EVS_ResetAllFiltersCmd_t`
- `typedef struct CFE_EVS_ResetFilterCmd CFE_EVS_ResetFilterCmd_t`
- `typedef struct CFE_EVS_DeleteEventFilterCmd CFE_EVS_DeleteEventFilterCmd_t`
- `typedef struct CFE_EVS_EnableAppEventTypeCmd CFE_EVS_EnableAppEventTypeCmd_t`
- `typedef struct CFE_EVS_DisableAppEventTypeCmd CFE_EVS_DisableAppEventTypeCmd_t`
- `typedef struct CFE_EVS_AddEventFilterCmd CFE_EVS_AddEventFilterCmd_t`
- `typedef struct CFE_EVS_SetFilterCmd CFE_EVS_SetFilterCmd_t`
- `typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t`
- `typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t`
- `typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t`

12.121.1 Detailed Description

Purpose: cFE Executive Services (EVS) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

12.121.2 Typedef Documentation

12.121.2.1 CFE_EVS_AddEventFilterCmd_t `typedef struct CFE_EVS_AddEventFilterCmd CFE_EVS_AddEventFilterCmd_t`

12.121.2.2 CFE_EVS_ClearLogCmd_t `typedef struct CFE_EVS_ClearLogCmd CFE_EVS_ClearLogCmd_t`

12.121.2.3 CFE_EVS_DeleteEventFilterCmd_t `typedef struct CFE_EVS_DeleteEventFilterCmd CFE_EVS_DeleteEventFilterCmd_t`

12.121.2.4 CFE_EVS_DisableAppEventsCmd_t `typedef struct CFE_EVS_DisableAppEventsCmd CFE_EVS_DisableAppEventsCmd_t`

12.121.2.5 CFE_EVS_DisableAppEventTypeCmd_t `typedef struct CFE_EVS_DisableAppEventTypeCmd CFE_EVS_DisableAppEventTypeCmd_t`

12.121.2.6 CFE_EVS_DisableEventTypeCmd_t `typedef struct CFE_EVS_DisableEventTypeCmd CFE_EVS_DisableEventTypeCmd_t`

12.121.2.7 CFE_EVS_DisablePortsCmd_t `typedef struct CFE_EVS_DisablePortsCmd CFE_EVS_DisablePortsCmd_t`

- 12.121.2.8 CFE_EVS_EnableAppEventsCmd_t** `typedef struct CFE_EVS_EnableAppEventsCmd CFE_EVS_EnableAppEventsCmd_t`
- 12.121.2.9 CFE_EVS_EnableAppEventTypeCmd_t** `typedef struct CFE_EVS_EnableAppEventTypeCmd CFE_EVS_EnableAppEventTypeCmd_t`
- 12.121.2.10 CFE_EVS_EnableEventTypeCmd_t** `typedef struct CFE_EVS_EnableEventTypeCmd CFE_EVS_EnableEventTypeCmd_t`
- 12.121.2.11 CFE_EVS_EnablePortsCmd_t** `typedef struct CFE_EVS_EnablePortsCmd CFE_EVS_EnablePortsCmd_t`
- 12.121.2.12 CFE_EVS_HousekeepingTlm_t** `typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t`
- 12.121.2.13 CFE_EVS_LongEventTlm_t** `typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t`
- 12.121.2.14 CFE_EVS_NoopCmd_t** `typedef struct CFE_EVS_NoopCmd CFE_EVS_NoopCmd_t`
- 12.121.2.15 CFE_EVS_ResetAllFiltersCmd_t** `typedef struct CFE_EVS_ResetAllFiltersCmd CFE_EVS_ResetAllFiltersCmd_t`
- 12.121.2.16 CFE_EVS_ResetAppCounterCmd_t** `typedef struct CFE_EVS_ResetAppCounterCmd CFE_EVS_ResetAppCounterCmd_t`
- 12.121.2.17 CFE_EVS_ResetCountersCmd_t** `typedef struct CFE_EVS_ResetCountersCmd CFE_EVS_ResetCountersCmd_t`
- 12.121.2.18 CFE_EVS_ResetFilterCmd_t** `typedef struct CFE_EVS_ResetFilterCmd CFE_EVS_ResetFilterCmd_t`
- 12.121.2.19 CFE_EVS_SendHkCmd_t** `typedef struct CFE_EVS_SendHkCmd CFE_EVS_SendHkCmd_t`
- 12.121.2.20 CFE_EVS_SetEventFormatModeCmd_t** `typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t`
Set Event Format Mode Command.
- 12.121.2.21 CFE_EVS_SetFilterCmd_t** `typedef struct CFE_EVS_SetFilterCmd CFE_EVS_SetFilterCmd_t`
- 12.121.2.22 CFE_EVS_SetLogModeCmd_t** `typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t`
Set Log Mode Command.
- 12.121.2.23 CFE_EVS_ShortEventTlm_t** `typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t`

12.121.2.24 CFE_EVS_WriteAppDataFileCmd_t `typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd`
Write Event Services Application Information to File Command.

12.121.2.25 CFE_EVS_WriteLogFileCmd_t `typedef struct CFE_EVS_WriteLogFileCmd CFE_EVS_WriteLogFileCmd`
Write Event Log to File Command.

12.122 cfe/modules/evs/config/default_cfe_evs_platform_cfg.h File Reference

```
#include "cfe_evs_mission_cfg.h"
#include "cfe_evs_internal_cfg.h"
```

12.122.1 Detailed Description

CFE Event Services (CFE_EVS) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.123 cfe/modules/evs/config/default_cfe_evs_topicids.h File Reference

Macros

- `#define CFE_MISSION_EVS_CMD_TOPICID 1`
- `#define CFE_MISSION_EVS_SEND_HK_TOPICID 9`
- `#define CFE_MISSION_EVS_HK_TLM_TOPICID 1`
- `#define CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID 8`
- `#define CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID 9`

12.123.1 Detailed Description

CFE Event Services (CFE_EVS) Application Topic IDs

12.123.2 Macro Definition Documentation

12.123.2.1 CFE_MISSION_EVS_CMD_TOPICID `#define CFE_MISSION_EVS_CMD_TOPICID 1`

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE EVS command messages

Limits

Not Applicable

Definition at line 35 of file default_cfe_evs_topicids.h.

12.123.2.2 CFE_MISSION_EVS_HK_TLM_TOPICID #define CFE_MISSION_EVS_HK_TLM_TOPICID 1

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE EVS telemetry messages

Limits

Not Applicable

Definition at line 47 of file default_cfe_evs_topicids.h.

12.123.2.3 CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID #define CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID 8

Definition at line 48 of file default_cfe_evs_topicids.h.

12.123.2.4 CFE_MISSION_EVS_SEND_HK_TOPICID #define CFE_MISSION_EVS_SEND_HK_TOPICID 9

Definition at line 36 of file default_cfe_evs_topicids.h.

12.123.2.5 CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID #define CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID 9

Definition at line 49 of file default_cfe_evs_topicids.h.

12.124 cfe/modules/evs/fsw/inc/cfe_evs_eventids.h File Reference

Macros

EVS event IDs

- #define **CFE_EVS_NOOP_EID** 0
EVS No-op Command Success Event ID.
- #define **CFE_EVS_STARTUP_EID** 1
EVS Initialization Event ID.
- #define **CFE_EVS_ERR_WRLOGFILE_EID** 2
EVS Write Event Log Command File Write Entry Failed Event ID.
- #define **CFE_EVS_ERR_CRLOGFILE_EID** 3
EVS Write Event Log Command Filename Parse or File Create Failed Event ID.
- #define **CFE_EVS_ERR_MSGID_EID** 5
EVS Invalid Message ID Received Event ID.
- #define **CFE_EVS_ERR_EVTIDNOREGS_EID** 6
EVS Command Event Not Registered For Filtering Event ID.
- #define **CFE_EVS_ERR_APPNOREGS_EID** 7
EVS Command Application Not Registered With EVS Event ID.
- #define **CFE_EVS_ERR_ILLAPPIDRANGE_EID** 8
EVS Command Get Application Data Failure Event ID.
- #define **CFE_EVS_ERR_NOAPPIDFOUND_EID** 9
EVS Command Get Application ID Failure Event ID.
- #define **CFE_EVS_ERR_ILLEGALFMTMOD_EID** 10
EVS Set Event Format Command Invalid Format Event ID.
- #define **CFE_EVS_ERR_MAXREGSFILTER_EID** 11

- #define CFE_EVS_ERR_WRDATFILE_EID 12
EVS Add Filter Command Max Filters Exceeded Event ID.
- #define CFE_EVS_ERR_CRDATFILE_EID 13
EVS Write Application Data Command Write Data Failure Event ID.
- #define CFE_EVS_WRITE_HEADER_ERR_EID 14
EVS Write Application Data Command Filename Parse or File Create Failed Event ID.
- #define CFE_EVS_ERR_CC_EID 15
EVS Invalid Command Code Received Event ID.
- #define CFE_EVS_RSTCNT_EID 16
EVS Reset Counters Command Success Event ID.
- #define CFE_EVS_SETFILTERMSK_EID 17
EVS Set Filter Command Success Event ID.
- #define CFE_EVS_ENAPORT_EID 18
EVS Enable Ports Command Success Event ID.
- #define CFE_EVS_DISPORT_EID 19
EVS Disable Ports Command Success Event ID.
- #define CFE_EVS_ENAEVTTYPE_EID 20
EVS Enable Event Type Command Success Event ID.
- #define CFE_EVS_DISEVTTYPE_EID 21
EVS Disable Event Type Command Success Event ID.
- #define CFE_EVS_SETEVTFMTMOD_EID 22
EVS Set Event Format Mode Command Success Event ID.
- #define CFE_EVS_ENAAPPEVTTYPE_EID 23
EVS Enable App Event Type Command Success Event ID.
- #define CFE_EVS_DISAPPENTTYPE_EID 24
EVS Disable App Event Type Command Success Event ID.
- #define CFE_EVS_ENAAPPEVNT_EID 25
EVS Enable App Events Command Success Event ID.
- #define CFE_EVS_DISAPPEVNT_EID 26
EVS Disable App Events Command Success Event ID.
- #define CFE_EVS_RSTEVCNT_EID 27
EVS Reset App Event Counter Command Success Event ID.
- #define CFE_EVS_RSTFILTER_EID 28
EVS Reset App Event Filter Command Success Event ID.
- #define CFE_EVS_RSTALLFILTER_EID 29
EVS Reset All Filters Command Success Event ID.
- #define CFE_EVS_ADDFILTER_EID 30
EVS Add Event Filter Command Success Event ID.
- #define CFE_EVS_DELFILTER_EID 31
EVS Delete Event Filter Command Success Event ID.
- #define CFE_EVS_WRDAT_EID 32
EVS Write Application Data Command Success Event ID.
- #define CFE_EVS_WRLOG_EID 33
EVS Write Event Log Command Success Event ID.
- #define CFE_EVS_EVT_FILTERED_EID 37
EVS Add Filter Command Duplicate Registration Event ID.
- #define CFE_EVS_LOGMODE_EID 38
EVS Set Log Mode Command Success Event ID.
- #define CFE_EVS_ERR_LOGMODE_EID 39
EVS Set Log Mode Command Invalid Mode Event ID.
- #define CFE_EVS_ERR_INVALID_BITMASK_EID 40
EVS Port Or Event Type Bitmask Invalid Event ID.
- #define CFE_EVS_ERR_UNREGISTERED_EVS_APP 41
EVS Send Event API App Not Registered With EVS Event ID.

- #define **CFE_EVS_FILTER_MAX_EID** 42
EVS Filter Max Count Reached Event ID.
- #define **CFE_EVS_LEN_ERR_EID** 43
EVS Invalid Command Length Event ID.
- #define **CFE_EVS_SQUELCHED_ERR_EID** 44
EVS Events Squelched Error Event ID.

12.124.1 Detailed Description

cFE Event Services Event IDs

12.124.2 Macro Definition Documentation

12.124.2.1 CFE_EVS_ADDFILTER_EID #define CFE_EVS_ADDFILTER_EID 30
EVS Add Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Add Event Filter Command](#) success.
Definition at line 367 of file cfe_evs_eventids.h.

12.124.2.2 CFE_EVS_DELFILTER_EID #define CFE_EVS_DELFILTER_EID 31
EVS Delete Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Delete Event Filter Command](#) success.
Definition at line 378 of file cfe_evs_eventids.h.

12.124.2.3 CFE_EVS_DISAPPENTTYPE_EID #define CFE_EVS_DISAPPENTTYPE_EID 24
EVS Disable App Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable App Event Type Command](#) success.
Definition at line 301 of file cfe_evs_eventids.h.

12.124.2.4 CFE_EVS_DISAPPEVT_EID #define CFE_EVS_DISAPPEVT_EID 26
EVS Disable App Events Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable App Events Command](#) success.
Definition at line 323 of file cfe_evs_eventids.h.

12.124.2.5 CFE_EVS_DISEVTTYPE_EID #define CFE_EVS_DISEVTTYPE_EID 21
EVS Disable Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable Event Type Command](#) success.
Definition at line 268 of file cfe_evs_eventids.h.

12.124.2.6 CFE_EVS_DISPORT_EID #define CFE_EVS_DISPORT_EID 19
EVS Disable Ports Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable Ports Command](#) success.
Definition at line 246 of file cfe_evs_eventids.h.

12.124.2.7 CFE_EVS_ENAAPPEVT_EID #define CFE_EVS_ENAAPPEVT_EID 25
EVS Enable App Events Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable App Events Command](#) success.
Definition at line 312 of file cfe_evs_eventids.h.

12.124.2.8 CFE_EVS_ENAAPPEVTTYPE_EID #define CFE_EVS_ENAAPPEVTTYPE_EID 23
EVS Enable App Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable App Event Type Command](#) success.
Definition at line 290 of file cfe_evs_eventids.h.

12.124.2.9 CFE_EVS_ENAEVTTYPE_EID #define CFE_EVS_ENAEVTTYPE_EID 20
EVS Enable Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable Event Type Command](#) success.
Definition at line 257 of file cfe_evs_eventids.h.

12.124.2.10 CFE_EVS_ENAPORT_EID #define CFE_EVS_ENAPORT_EID 18
EVS Enable Ports Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable Ports Command](#) success.
Definition at line 235 of file cfe_evs_eventids.h.

12.124.2.11 CFE_EVS_ERR_APPNOREGS_EID #define CFE_EVS_ERR_APPNOREGS_EID 7
EVS Command Application Not Registered With EVS Event ID.

Type: ERROR

Cause:

An EVS command handler failure due to the referenced application not being registered with EVS. OVERLOADED
Definition at line 110 of file cfe_evs_eventids.h.

12.124.2.12 CFE_EVS_ERR_CC_EID #define CFE_EVS_ERR_CC_EID 15
EVS Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE_EVS_CMD_MID](#) received on the EVS message pipe.
Definition at line 202 of file cfe_evs_eventids.h.

12.124.2.13 CFE_EVS_ERR_CRDATFILE_EID #define CFE_EVS_ERR_CRDATFILE_EID 13
EVS Write Application Data Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[Write Application Data Command](#) failed to parse the filename or open/create the file. OVERLOADED
Definition at line 180 of file cfe_evs_eventids.h.

12.124.2.14 CFE_EVS_ERR_CRLOGFILE_EID #define CFE_EVS_ERR_CRLOGFILE_EID 3
EVS Write Event Log Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[EVS Write Event Log Command](#) failure parsing the file name or during open/creation of the file. OVERLOADED
Definition at line 77 of file cfe_evs_eventids.h.

12.124.2.15 CFE_EVS_ERR_EVTIDNOREGS_EID #define CFE_EVS_ERR_EVTIDNOREGS_EID 6
EVS Command Event Not Registered For Filtering Event ID.

Type: ERROR

Cause:

An EVS command handler failure due to the event not being registered for filtering. OVERLOADED
Definition at line 99 of file cfe_evs_eventids.h.

12.124.2.16 CFE_EVS_ERR_ILLAPPIDRANGE_EID #define CFE_EVS_ERR_ILLAPPIDRANGE_EID 8
EVS Command Get Application Data Failure Event ID.

Type: ERROR

Cause:

An EVS command handler failure retrieving the application data. OVERLOADED
Definition at line 121 of file cfe_evs_eventids.h.

12.124.2.17 CFE_EVS_ERR_ILLEGALFMTMOD_EID #define CFE_EVS_ERR_ILLEGALFMTMOD_EID 10
EVS Set Event Format Command Invalid Format Event ID.

Type: ERROR

Cause:

[EVS Set Event Format Command](#) failure due to invalid format argument.
Definition at line 144 of file cfe_evs_eventids.h.

12.124.2.18 CFE_EVS_ERR_INVALID_BITMASK_EID #define CFE_EVS_ERR_INVALID_BITMASK_EID 40
EVS Port Or Event Type Bitmask Invalid Event ID.

Type: ERROR

Cause:

Invalid bitmask for EVS port or event type. OVERLOADED
Definition at line 446 of file cfe_evs_eventids.h.

12.124.2.19 CFE_EVS_ERR_LOGMODE_EID #define CFE_EVS_ERR_LOGMODE_EID 39
EVS Set Log Mode Command Invalid Mode Event ID.

Type: ERROR

Cause:

[EVS Set Log Mode Command](#) failure due to invalid log mode.
Definition at line 435 of file cfe_evs_eventids.h.

12.124.2.20 CFE_EVS_ERR_MAXREGSFILTER_EID #define CFE_EVS_ERR_MAXREGSFILTER_EID 11
EVS Add Filter Command Max Filters Exceeded Event ID.

Type: ERROR

Cause:

[EVS Add Filter Command](#) failure due to exceeding the maximum number of filters.
Definition at line 156 of file cfe_evs_eventids.h.

12.124.2.21 CFE_EVS_ERR_MSGID_EID #define CFE_EVS_ERR_MSGID_EID 5
EVS Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the EVS message pipe.
Definition at line 88 of file cfe_evs_eventids.h.

12.124.2.22 CFE_EVS_ERR_NOAPPIDFOUND_EID #define CFE_EVS_ERR_NOAPPIDFOUND_EID 9
EVS Command Get Application ID Failure Event ID.

Type: ERROR

Cause:

An EVS command handler failure retrieving the application ID. OVERLOADED
Definition at line 132 of file cfe_evs_eventids.h.

12.124.2.23 CFE_EVS_ERR_UNREGISTERED_EVS_APP #define CFE_EVS_ERR_UNREGISTERED_EVS_APP 41
EVS Send Event API App Not Registered With EVS Event ID.

Type: ERROR

Cause:

An EVS Send Event API called for application not registered with EVS.
Definition at line 457 of file cfe_evs_eventids.h.

12.124.2.24 CFE_EVS_ERR_WRDATFILE_EID #define CFE_EVS_ERR_WRDATFILE_EID 12
EVS Write Application Data Command Write Data Failure Event ID.

Type: ERROR

Cause:

[Write Application Data Command](#) failure to write application EVS data.
Definition at line 168 of file cfe_evs_eventids.h.

12.124.2.25 CFE_EVS_ERR_WRLlogfile_EID #define CFE_EVS_ERR_WRLlogfile_EID 2
EVS Write Event Log Command File Write Entry Failed Event ID.

Type: ERROR

Cause:

[EVS Write Event Log Command](#) failure writing data to the file.
Definition at line 65 of file cfe_evs_eventids.h.

12.124.2.26 CFE_EVS_EVT_FILTERED_EID #define CFE_EVS_EVT_FILTERED_EID 37
EVS Add Filter Command Duplicate Registration Event ID.

Type: ERROR

Cause:

[EVS Add Filter Command](#) failure due to event already being registered for filtering.
Definition at line 412 of file cfe_evs_eventids.h.

12.124.2.27 CFE_EVS_FILTER_MAX_EID #define CFE_EVS_FILTER_MAX_EID 42
EVS Filter Max Count Reached Event ID.

Type: INFORMATIONAL

Cause:

Filter count for the event reached CFE_EVS_MAX_FILTER_COUNT and is latched until filter is reset.
Definition at line 468 of file cfe_evs_eventids.h.

12.124.2.28 CFE_EVS_LEN_ERR_EID #define CFE_EVS_LEN_ERR_EID 43
EVS Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE_EVS_CMD_MID](#) received on the EVS message pipe.
Definition at line 479 of file cfe_evs_eventids.h.

12.124.2.29 CFE_EVS_LOGMODE_EID #define CFE_EVS_LOGMODE_EID 38
EVS Set Log Mode Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Log Mode Command](#) success.
Definition at line 423 of file cfe_evs_eventids.h.

12.124.2.30 CFE_EVS_NOOP_EID #define CFE_EVS_NOOP_EID 0
EVS No-op Command Success Event ID.

Type: INFORMATION

Cause:

[EVS NO-OP command](#) success.
Definition at line 42 of file cfe_evs_eventids.h.

12.124.2.31 CFE_EVS_RSTALLFILTER_EID #define CFE_EVS_RSTALLFILTER_EID 29
EVS Reset All Filters Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset All Filters Command](#) success.
Definition at line 356 of file cfe_evs_eventids.h.

12.124.2.32 CFE_EVS_RSTCNT_EID #define CFE_EVS_RSTCNT_EID 16
EVS Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset Counters Command](#) success.
Definition at line 213 of file cfe_evs_eventids.h.

12.124.2.33 CFE_EVS_RSTEVTCNT_EID #define CFE_EVS_RSTEVTCNT_EID 27
EVS Reset App Event Counter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset App Event Counter Command](#) success.
Definition at line 334 of file cfe_evs_eventids.h.

12.124.2.34 CFE_EVS_RSTFILTER_EID #define CFE_EVS_RSTFILTER_EID 28
EVS Reset App Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset App Event Filter Command](#) success.
Definition at line 345 of file cfe_evs_eventids.h.

12.124.2.35 CFE_EVS_SETEVTFMTMOD_EID #define CFE_EVS_SETEVTFMTMOD_EID 22
EVS Set Event Format Mode Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Event Format Mode Command](#) success.
Definition at line 279 of file cfe_evs_eventids.h.

12.124.2.36 CFE_EVS_SETFILTERMSK_EID #define CFE_EVS_SETFILTERMSK_EID 17
EVS Set Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Filter Command](#) success.

Definition at line 224 of file cfe_evs_eventids.h.

12.124.2.37 CFE_EVS_SQUELCHED_ERR_EID #define CFE_EVS_SQUELCHED_ERR_EID 44
EVS Events Squelched Error Event ID.

Type: ERROR

Cause:

Events generated in app at a rate in excess of [CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST](#) in one moment or [CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC](#) sustained

Definition at line 492 of file cfe_evs_eventids.h.

12.124.2.38 CFE_EVS_STARTUP_EID #define CFE_EVS_STARTUP_EID 1
EVS Initialization Event ID.

Type: INFORMATION

Cause:

Event Services Task initialization complete.

Definition at line 53 of file cfe_evs_eventids.h.

12.124.2.39 CFE_EVS_WRDAT_EID #define CFE_EVS_WRDAT_EID 32
EVS Write Application Data Command Success Event ID.

Type: DEBUG

Cause:

[EVS Write Application Data Command](#) success.

Definition at line 389 of file cfe_evs_eventids.h.

12.124.2.40 CFE_EVS_WRITE_HEADER_ERR_EID #define CFE_EVS_WRITE_HEADER_ERR_EID 14
EVS Write File Header to Log File Failure Event ID.

Type: ERROR

Cause:

Bytes written during Write File Header to Log File was not equal to the expected header size.
Definition at line 191 of file cfe_evs_eventids.h.

12.124.2.41 CFE_EVS_WRLOG_EID #define CFE_EVS_WRLOG_EID 33
EVS Write Event Log Command Success Event ID.

Type: DEBUG

Cause:

EVS Write Event Log Command success.
Definition at line 400 of file cfe_evs_eventids.h.

12.125 cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_fs_filedef.h"
```

12.125.1 Detailed Description

Declarations and prototypes for cfe_fs_extern_typedefs module

12.126 cfe/modules/fs/config/default_cfe_fs_filedef.h File Reference

```
#include "common_types.h"
#include "cfe_fs_interface_cfg.h"
```

Data Structures

- struct [CFE_FS_Header](#)
Standard cFE File header structure definition.

Typedefs

- typedef uint32 [CFE_FS_SubType_Enum_t](#)
Content descriptor for File Headers.
- typedef struct [CFE_FS_Header](#) [CFE_FS_Header_t](#)
Standard cFE File header structure definition.

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_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_PIPE DATA` = 20, `CFE_FS_SubType_SB_ROUTEDATA` = 21, `CFE_FS_SubType_SB_MAPDATA` = 22,
 `CFE_FS_SubType_ES_QUERYALLTASKS` = 23 }

File subtypes used within cFE.

12.126.1 Detailed Description

Declarations and prototypes for `cfe_fs_extern_typedefs` module

12.126.2 Typedef Documentation

12.126.2.1 `CFE_FS_Header_t` `typedef struct CFE_FS_Header CFE_FS_Header_t`

Standard cFE File header structure definition.

12.126.2.2 `CFE_FS_SubType_Enum_t` `typedef uint32 CFE_FS_SubType_Enum_t`

Content descriptor for File Headers.

See also

enum `CFE_FS_SubType`

Definition at line 176 of file `default_cfe_fs_filedef.h`.

12.126.3 Enumeration Type Documentation

12.126.3.1 `CFE_FS_SubType` `enum CFE_FS_SubType`

File subtypes used within cFE.

This defines all the file subtypes used by cFE. Note apps can extend as needed but need to avoid conflicts (app context not currently included in the file header).

Enumerator

<code>CFE_FS_SubType_ES_ERLOG</code>	Executive Services Exception/Reset Log Type. Executive Services Exception/Reset Log File which is generated in response to a <code>\$sc_\$cpu_ES_WriteERLog2File</code> command.
<code>CFE_FS_SubType_ES_SYSLOG</code>	Executive Services System Log Type. Executive Services System Log File which is generated in response to a <code>\$sc_\$cpu_ES_WriteSysLog2File</code> command.
<code>CFE_FS_SubType_ES_QUERYALL</code>	Executive Services Information on All Applications File. Executive Services Information on All Applications File which is generated in response to a <code>\$sc_\$cpu_ES_WriteAppInfo2File</code> command.

Enumerator

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_StopLData 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_PIPE DATA	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 39 of file default_cfe_fs_filedef.h.

12.127 cfe/modules/fs/config/default_cfe_fs_interface_cfg.h File Reference

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')

12.127.1 Detailed Description

Declarations and prototypes for cfe_fs_extern_typedefs module

12.127.2 Macro Definition Documentation

12.127.2.1 CFE_FS_FILE_CONTENT_ID #define CFE_FS_FILE_CONTENT_ID 0x63464531
Magic Number for cFE compliant files (= 'cFE1')
Definition at line 39 of file default_cfe_fs_interface_cfg.h.

12.127.2.2 CFE_FS_HDR_DESC_MAX_LEN #define CFE_FS_HDR_DESC_MAX_LEN 32
Max length of description field in a standard cFE File Header.
Definition at line 37 of file default_cfe_fs_interface_cfg.h.

12.128 cfe/modules/fs/config/default_cfe_fs_mission_cfg.h File Reference

```
#include "cfe_fs_interface_cfg.h"
```

12.128.1 Detailed Description

CFE File Services (CFE_FS) Application Mission Configuration Header File
This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.129 cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference

```
#include "common_types.h"
```

Data Structures

- struct [CCSDS_PrimaryHeader](#)
CCSDS packet primary header.
- struct [CCSDS_ExtendedHeader](#)
CCSDS packet extended header.

Typedefs

- typedef struct [CCSDS_PrimaryHeader](#) CCSDS_PrimaryHeader_t
CCSDS packet primary header.
- typedef struct [CCSDS_ExtendedHeader](#) CCSDS_ExtendedHeader_t
CCSDS packet extended header.

12.129.1 Detailed Description

Define CCSDS packet header types

- Avoid direct access for portability, use APIs
- Used to construct message structures

12.129.2 Typedef Documentation

12.129.2.1 CCSDS_ExtendedHeader_t `typedef struct CCSDS_ExtendedHeader CCSDS_ExtendedHeader_t`
 CCSDS packet extended header.

12.129.2.2 CCSDS_PrimaryHeader_t `typedef struct CCSDS_PrimaryHeader CCSDS_PrimaryHeader_t`
 CCSDS packet primary header.

12.130 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference

```
#include "cfe_resourceid_basevalue.h"
```

Enumerations

- enum {
`CFE_RESOURCEID_ES_TASKID_BASE_OFFSET = OS_OBJECT_TYPE_OS_TASK, CFE_RESOURCEID_ES_APPID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 1, CFE_RESOURCEID_ES_LIBID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 2, CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 3, CFE_RESOURCEID_ES_POOLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 4, CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 5, CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET = OS_OBJECT_TYPE_USER + 6, CFE_RESOURCEID_CONFIGID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 7, CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 8, CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 9 }`
- enum {
`CFE_ES_TASKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_OFFSET), CFE_ES_APPID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BASE_OFFSET), CFE_ES_LIBID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_LIBID_BASE_OFFSET), CFE_ES_COUNTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET), CFE_ES_POOLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_OFFSET), CFE_ES_CDSBLOCKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET), CFE_SB_PIPEID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET), CFE_CONFIGID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_CONFIGID_BASE_OFFSET), CFE_TBL_VALRESULTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET), CFE_TBL_DUMPCTRLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET) }`

12.130.1 Detailed Description

Contains CFE internal prototypes and definitions related to resource management and related CFE resource IDs. A CFE ES Resource ID is a common way to identify CFE-managed resources such as apps, tasks, counters, memory pools, CDS blocks, and other entities.

12.131 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference

```
#include "cfe_resourceid_typedef.h"
#include "osapi-idmap.h"
```

Macros

- `#define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT`

- #define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK
- #define CFE_RESOURCEID_MAKE_BASE(offset) (CFE_RESOURCEID_MARK | ((offset) << CFE_RESOURCEID_SHIFT))
A macro to generate a CFE resource ID base value from an offset.

12.131.1 Detailed Description

An implementation of CFE resource ID base values/limits that will be compatible with OSAL IDs. This is intended as a transitional tool to provide runtime value uniqueness, particularly when the "simple" (compatible) resource ID implementation is used. In this mode, compiler type checking is disabled, and so OSAL IDs can be silently interchanged with CFE IDs.

However, by ensuring uniqueness in the runtime values, any ID handling errors may at least be detectable at runtime. This still works fine with the "strict" resource ID option, but is less important as the compiler type checking should prevent this type of error before the code even runs.

The downside to this implementation is that it has a dependency on the OSAL ID structure.

12.131.2 Macro Definition Documentation

12.131.2.1 CFE_RESOURCEID_MAKE_BASE #define CFE_RESOURCEID_MAKE_BASE (offset) (CFE_RESOURCEID_MARK | ((offset) << CFE_RESOURCEID_SHIFT))

A macro to generate a CFE resource ID base value from an offset.

Each CFE ID range is effectively an extension of OSAL ID ranges by starting at OS_OBJECT_TYPE_USER.
Definition at line 73 of file cfe_resourceid_basevalue.h.

12.131.2.2 CFE_RESOURCEID_MAX #define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK Definition at line 65 of file cfe_resourceid_basevalue.h.

12.131.2.3 CFE_RESOURCEID_SHIFT #define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT Definition at line 64 of file cfe_resourceid_basevalue.h.

12.132 cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_resourceid_typedef.h"
```

Data Structures

- struct CFE_SB_MsgId_t
CFE_SB_MsgId_t type definition.
- struct CFE_SB_Qos_t
Quality Of Service Type Definition.

Macros

- #define CFE_SB_SUB_ENTRIES_PER_PKT 20
Configuration parameter used by SBN App.

Typedefs

- `typedef uint8 CFE_SB_QosPriority_Enum_t`
Selects the priority level for message routing.
- `typedef uint8 CFE_SB_QosReliability_Enum_t`
Selects the reliability level for message routing.
- `typedef uint16 CFE_SB_Routeld_Atom_t`
An integer type that should be used for indexing into the Routing Table.
- `typedef uint32 CFE_SB_MsgId_Atom_t`
CFE_SB_MsgId_Atom_t primitive type definition.
- `typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t`
CFE_SB_PipeId_t to primitive 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.

12.132.1 Detailed Description

Declarations and prototypes for cfe_sb_extern_typedefs module

12.132.2 Macro Definition Documentation

12.132.2.1 CFE_SB_SUB_ENTRIES_PER_PKT `#define CFE_SB_SUB_ENTRIES_PER_PKT 20`
 Configuration parameter used by SBN App.
 Definition at line 32 of file default_cfe_sb_extern_typedefs.h.

12.132.3 Typedef Documentation

12.132.3.1 CFE_SB_MsgId_Atom_t `typedef uint32 CFE_SB_MsgId_Atom_t`
`CFE_SB_MsgId_Atom_t` primitive type definition.

This is an integer type capable of holding any Message ID value Note: This value is limited via `CFE_PLATFORM_SB_HIGHEST_VALID_MESSAGE_ID`.
 Definition at line 91 of file default_cfe_sb_extern_typedefs.h.

12.132.3.2 CFE_SB_PipeId_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t`
`CFE_SB_PipeId_t` to primitive type definition.
 Software Bus pipe identifier used in many SB APIs, as well as SB Telemetry messages and data files.
 Definition at line 114 of file default_cfe_sb_extern_typedefs.h.

12.132.3.3 CFE_SB_QosPriority_Enum_t `typedef uint8 CFE_SB_QosPriority_Enum_t`
Selects the priority level for message routing.

See also

enum [CFE_SB_QosPriority](#)

Definition at line 55 of file default_cfe_sb_extern_typedefs.h.

12.132.3.4 CFE_SB_QosReliability_Enum_t `typedef uint8 CFE_SB_QosReliability_Enum_t`
Selects the reliability level for message routing.

See also

enum [CFE_SB_QosReliability](#)

Definition at line 78 of file default_cfe_sb_extern_typedefs.h.

12.132.3.5 CFE_SB_RouteId_Atom_t `typedef uint16 CFE_SB_RouteId_Atom_t`

An integer type that should be used for indexing into the Routing Table.

Definition at line 83 of file default_cfe_sb_extern_typedefs.h.

12.132.4 Enumeration Type Documentation

12.132.4.1 CFE_SB_QosPriority `enum CFE_SB_QosPriority`

Label definitions associated with CFE_SB_QosPriority_Enum_t.

Enumerator

<code>CFE_SB_QosPriority_LOW</code>	Normal priority level.
<code>CFE_SB_QosPriority_HIGH</code>	High priority.

Definition at line 37 of file default_cfe_sb_extern_typedefs.h.

12.132.4.2 CFE_SB_QosReliability `enum CFE_SB_QosReliability`

Label definitions associated with CFE_SB_QosReliability_Enum_t.

Enumerator

<code>CFE_SB_QosReliability_LOW</code>	Normal (best-effort) reliability.
<code>CFE_SB_QosReliability_HIGH</code>	High reliability.

Definition at line 60 of file default_cfe_sb_extern_typedefs.h.

12.133 cfe/modules/sb/config/default_cfe_sb_fcncodes.h File Reference

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_WRITE_ROUTING_INFO_CC 3
- #define CFE_SB_ENABLE_ROUTE_CC 4
- #define CFE_SB_DISABLE_ROUTE_CC 5
- #define CFE_SB_WRITE_PIPE_INFO_CC 7
- #define CFE_SB_WRITE_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

12.133.1 Detailed Description

Specification for the CFE Event Services (CFE_SB) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.133.2 Macro Definition Documentation

12.133.2.1 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 parameters in the command. All destinations are enabled by default.

Command Mnemonic(s) \$sc_\$cpu_SB_DisRoute

Command Structure

CFE_SB_DisableRouteCmd_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_WRITE_ROUTING_INFO_CC to verify enable/disable state change
- The CFE_SB_DSBL RTE2_EID debug event message will be generated
- Destination will stop receiving messages

Error Conditions

This command may fail for the following reason(s):

- the MsgId or PipeId parameters do not pass validation
- 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 resetting 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.

Definition at line 271 of file default_cfe_sb_fcncodes.h.

12.133.2.2 CFE_SB_DISABLE_SUB_REPORTING_CC #define CFE_SB_DISABLE_SUB_REPORTING_CC 10**Name** Disable Subscription Reporting Command**Description**

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 needed by SBN if offboard routing is required.

Command Mnemonic(s) \$sc_\$cpu_SB_DisSubRptg**Command Structure**

[CFE_SB_DisableSubReportingCmd_t](#)

Command Verification

Successful execution of this command will result in the suppression of packets (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_ENABLE_SUB_REPORTING_CC](#), [CFE_SB_SEND_PREV_SUBS_CC](#)

Definition at line 428 of file default_cfe_sb_fcncodes.h.

12.133.2.3 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 parameters 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_EnableRouteCmd_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_WRITE_ROUTING_INFO_CC](#) to verify enable/disable state change
- The [CFE_SB_ENBL RTE2_EID](#) debug event message will be generated
- Destination will begin receiving messages

Error Conditions

This command may fail for the following reason(s):

- the MsgId or PipeId parameters do not pass validation
- 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.

Definition at line 230 of file default_cfe_sb_fcncodes.h.

12.133.2.4 CFE_SB_ENABLE_SUB_REPORTING_CC #define CFE_SB_ENABLE_SUB_REPORTING_CC 9

Name Enable Subscription Reporting Command

Description

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 needed by SBN if offboard routing is required.

Command Mnemonic(s) \$sc_\$cpu_SB_EnaSubRptg

Command Structure

[CFE_SB_EnableSubReportingCmd_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 395 of file default_cfe_sb_fcncodes.h.

12.133.2.5 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_NoopCmd_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 66 of file default_cfe_sb_fcncodes.h.

12.133.2.6 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)
- No Subscribers Counter (\$sc_\$cpu_SB_NoSubEC)
- Duplicate Subscriptions Counter (\$sc_\$cpu_SB_DupSubCnt)
- Msg Send Error Counter (\$sc_\$cpu_SB_MsgSndEC)
- Msg Receive Error Counter (\$sc_\$cpu_SB_MsgRecEC)
- Internal Error Counter (\$sc_\$cpu_SB_InternalEC)
- Create Pipe Error Counter (\$sc_\$cpu_SB_NewPipeEC)
- Subscribe Error Counter (\$sc_\$cpu_SB_SubscrEC)
- Pipe Overflow Error Counter (\$sc_\$cpu_SB_PipeOvrEC)
- Msg Limit Error Counter (\$sc_\$cpu_SB_MsgLimEC)

Command Mnemonic(s) \$sc_\$cpu_SB_ResetCtrs**Command Structure**[CFE_SB_ResetCountersCmd_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_SB_CMDPC** - command execution counter will be reset to 0
- All other counters listed in description will be reset to 0
- 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 113 of file default_cfe_sb_fcncodes.h.

12.133.2.7 CFE_SB_SEND_PREV_SUBS_CC #define CFE_SB_SEND_PREV_SUBS_CC 11**Name** Send Previous Subscriptions Command**Description** 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 SBN(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_SendPrevSubsCmd_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_REPORTING_CC](#)

Definition at line 460 of file default_cfe_sb_fcncodes.h.

12.133.2.8 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_SendSbStatsCmd_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

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 147 of file default_cfe_sb_fcncodes.h.

12.133.2.9 CFE_SB_WRITE_MAP_INFO_CC #define CFE_SB_WRITE_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 software bus send operation. This is diagnostic information that may be needed due to the dynamic nature of the cFE software bus. An absolute 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_MAP_FILENAME](#).

Command Mnemonic(s) \$sc_\$cpu_SB_WriteMap2File

Command Structure

[CFE_SB_WriteMapInfoCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_SB_CMDPC** - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE_SB SND RTG EID](#) debug event message will be generated

Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

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.

Definition at line 362 of file default_cfe_sb_fcncodes.h.

12.133.2.10 CFE_SB_WRITE_PIPE_INFO_CC #define CFE_SB_WRITE_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 absolute 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_WritePipeInfoCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_SB_CMDPC` - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE_SB SND RTG_EID](#) debug event message will be generated

Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

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.

Definition at line 316 of file default_cfe_sb_fcncodes.h.

12.133.2.11 CFE_SB_WRITE_ROUTING_INFO_CC #define CFE_SB_WRITE_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 absolute 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_WriteRoutingInfoCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_SB_CMDPC` - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE_SB SND RTG_EID](#) debug event message will be generated

Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

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.

Definition at line 192 of file default_cfe_sb_fcncodes.h.

12.134 cfe/modules/sb/config/default_cfe_sb_interface_cfg.h File Reference

Macros

- `#define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768`
- `#define CFE_MISSION_SB_MAX_PIPES 64`

12.134.1 Detailed Description

CFE Software Bus (CFE_SB) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.134.2 Macro Definition Documentation

12.134.2.1 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 definitions 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 67 of file default_cfe_sb_interface_cfg.h.

12.134.2.2 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 50 of file default_cfe_sb_interface_cfg.h.

12.135 cfe/modules/sb/config/default_cfe_sb_internal_cfg.h File Reference

Macros

- #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_HIGHEST_VALID_MSGID 0x1FFF
- #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_SUBSCRIPTION_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 + 128)
- #define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
- #define CFE_PLATFORM_SB_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE

12.135.1 Detailed Description

CFE Software Bus (CFE_SB) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.135.2 Macro Definition Documentation

12.135.2.1 CFE_PLATFORM_SB_BUF_MEMORY_BYTES #define CFE_PLATFORM_SB_BUF_MEMORY_BYT←
ES 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 123 of file default_cfe_sb_internal_cfg.h.

12.135.2.2 CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME #define CFE_PLATFORM_SB_DEFAULT_MAP_FILE←
NAME "/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 193 of file default_cfe_sb_internal_cfg.h.

12.135.2.3 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 101 of file default_cfe_sb_internal_cfg.h.

12.135.2.4 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME #define CFE_PLATFORM_SB_DEFAULT_PIPE_FIL←
ENAME "/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 176 of file default_cfe_sb_internal_cfg.h.

12.135.2.5 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 162 of file default_cfe_sb_internal_cfg.h.

12.135.2.6 CFE_PLATFORM_SB_FILTER_MASK1 #define CFE_PLATFORM_SB_FILTER_MASK1 CFE_EVS_FIRST_4_STOP
Definition at line 211 of file default_cfe_sb_internal_cfg.h.

12.135.2.7 CFE_PLATFORM_SB_FILTER_MASK2 #define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
Definition at line 214 of file default_cfe_sb_internal_cfg.h.

12.135.2.8 CFE_PLATFORM_SB_FILTER_MASK3 #define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
Definition at line 217 of file default_cfe_sb_internal_cfg.h.

12.135.2.9 CFE_PLATFORM_SB_FILTER_MASK4 #define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
Definition at line 220 of file default_cfe_sb_internal_cfg.h.

12.135.2.10 CFE_PLATFORM_SB_FILTER_MASK5 #define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
Definition at line 223 of file default_cfe_sb_internal_cfg.h.

12.135.2.11 CFE_PLATFORM_SB_FILTER_MASK6 #define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
Definition at line 226 of file default_cfe_sb_internal_cfg.h.

12.135.2.12 CFE_PLATFORM_SB_FILTER_MASK7 #define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
Definition at line 229 of file default_cfe_sb_internal_cfg.h.

12.135.2.13 CFE_PLATFORM_SB_FILTER_MASK8 #define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
Definition at line 232 of file default_cfe_sb_internal_cfg.h.

12.135.2.14 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 parameters dictates what SB events will be filtered through SB. The filtering will begin after the SB task initializes and stay in effect until a cmd to SB 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 210 of file default_cfe_sb_internal_cfg.h.

12.135.2.15 CFE_PLATFORM_SB_FILTERED_EVENT2

```
#define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
```

Definition at line 213 of file default_cfe_sb_internal_cfg.h.

12.135.2.16 CFE_PLATFORM_SB_FILTERED_EVENT3

```
#define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
```

Definition at line 216 of file default_cfe_sb_internal_cfg.h.

12.135.2.17 CFE_PLATFORM_SB_FILTERED_EVENT4

```
#define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
```

Definition at line 219 of file default_cfe_sb_internal_cfg.h.

12.135.2.18 CFE_PLATFORM_SB_FILTERED_EVENT5

```
#define CFE_PLATFORM_SB_FILTERED_EVENT5 0
```

Definition at line 222 of file default_cfe_sb_internal_cfg.h.

12.135.2.19 CFE_PLATFORM_SB_FILTERED_EVENT6

```
#define CFE_PLATFORM_SB_FILTERED_EVENT6 0
```

Definition at line 225 of file default_cfe_sb_internal_cfg.h.

12.135.2.20 CFE_PLATFORM_SB_FILTERED_EVENT7

```
#define CFE_PLATFORM_SB_FILTERED_EVENT7 0
```

Definition at line 228 of file default_cfe_sb_internal_cfg.h.

12.135.2.21 CFE_PLATFORM_SB_FILTERED_EVENT8

```
#define CFE_PLATFORM_SB_FILTERED_EVENT8 0
```

Definition at line 231 of file default_cfe_sb_internal_cfg.h.

12.135.2.22 CFE_PLATFORM_SB_HIGHEST_VALID_MSGID

```
#define CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
```

ID 0x1FFF

Purpose Highest Valid Message Id

Description:

The value of this constant dictates the range of valid message ID's, from 0 to CFE_PLATFORM_SB_HIGHEST_VALID_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

Limits

This parameter has a lower limit is 1, and an upper limit of 0xFFFFFFFF.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE_PLATFORM_SB_MAX_MSG_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 148 of file default_cfe_sb_internal_cfg.h.

12.135.2.23 CFE_PLATFORM_SB_MAX_BLOCK_SIZE #define CFE_PLATFORM_SB_MAX_BLOCK_SIZE ([CFE_MISSION_SB_MAX_SB_M](#)
+ 128)

Definition at line 261 of file default_cfe_sb_internal_cfg.h.

12.135.2.24 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 86 of file default_cfe_sb_internal_cfg.h.

12.135.2.25 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 effect 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 must be a power of two if software bus message routing hash implementation is being used. Lower than 64 will cause unit test failures, and telemetry reporting is impacted below 32. There is no hard upper limit, but impacts memory footprint. For software bus message routing search implementation the number of msg ids subscribed to impacts performance.

Definition at line 53 of file default_cfe_sb_internal_cfg.h.

12.135.2.26 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 effect 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 70 of file default_cfe_sb_internal_cfg.h.

12.135.2.27 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_PLATFORM_ES_POOL_MAX_BUCKETS](#)

Definition at line 245 of file default_cfe_sb_internal_cfg.h.

12.135.2.28 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16

Definition at line 246 of file default_cfe_sb_internal_cfg.h.

12.135.2.29 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20

Definition at line 247 of file default_cfe_sb_internal_cfg.h.

12.135.2.30 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36

Definition at line 248 of file default_cfe_sb_internal_cfg.h.

12.135.2.31 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64

Definition at line 249 of file default_cfe_sb_internal_cfg.h.

12.135.2.32 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96

Definition at line 250 of file default_cfe_sb_internal_cfg.h.

12.135.2.33 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
07 128

Definition at line 251 of file default_cfe_sb_internal_cfg.h.

12.135.2.34 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
08 160

Definition at line 252 of file default_cfe_sb_internal_cfg.h.

12.135.2.35 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
09 256

Definition at line 253 of file default_cfe_sb_internal_cfg.h.

12.135.2.36 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
10 512

Definition at line 254 of file default_cfe_sb_internal_cfg.h.

12.135.2.37 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
11 1024

Definition at line 255 of file default_cfe_sb_internal_cfg.h.

12.135.2.38 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
12 2048

Definition at line 256 of file default_cfe_sb_internal_cfg.h.

12.135.2.39 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
13 4096

Definition at line 257 of file default_cfe_sb_internal_cfg.h.

12.135.2.40 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
14 8192

Definition at line 258 of file default_cfe_sb_internal_cfg.h.

12.135.2.41 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
15 16384

Definition at line 259 of file default_cfe_sb_internal_cfg.h.

12.135.2.42 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
16 32768

Definition at line 260 of file default_cfe_sb_internal_cfg.h.

12.135.2.43 CFE_PLATFORM_SB_START_TASK_PRIORITY #define CFE_PLATFORM_SB_START_TASK_PRIORITY←
TY 64

Purpose Define SB Task Priority

Description:

Defines the cFE_SB Task priority.

Limits

Not Applicable

Definition at line 272 of file default_cfe_sb_internal_cfg.h.

12.135.2.44 CFE_PLATFORM_SB_START_TASK_STACK_SIZE #define CFE_PLATFORM_SB_START_TASK_STA←
CK_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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 287 of file default_cfe_sb_internal_cfg.h.

12.136 cfe/modules/sb/config/default_cfe_sb_mission_cfg.h File Reference

```
#include "cfe_sb_interface_cfg.h"
```

12.136.1 Detailed Description

CFE Event Services (CFE_SB) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.137 cfe/modules/sb/config/default_cfe_sb_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_sb_fcncodes.h"
#include "cfe_sb_msgdefs.h"
#include "cfe_sb_msgstruct.h"
```

12.137.1 Detailed Description

Specification for the CFE Event Services (CFE_SB) command and telemetry message data types.
This is a compatibility header for the "cfe_sb_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.138 cfe/modules/sb/config/default_cfe_sb_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_sb_extern_typedefs.h"
#include "cfe_sb_fcncodes.h"
```

Data Structures

- struct [CFE_SB_WriteFileInfoCmd_Payload](#)
Write File Info Command Payload.
- struct [CFE_SB_RouteCmd_Payload](#)
Enable/Disable Route Command Payload.
- struct [CFE_SB_HousekeepingTlm_Payload](#)
- struct [CFE_SB_PipeDepthStats](#)
SB Pipe Depth Statistics.
- struct [CFE_SB_PipeInfoEntry](#)
SB Pipe Information File Entry.
- struct [CFE_SB_StatsTlm_Payload](#)
- struct [CFE_SB_RoutingFileEntry](#)
SB Routing File Entry.
- struct [CFE_SB_MsgMapFileEntry](#)
SB Map File Entry.
- struct [CFE_SB_SingleSubscriptionTlm_Payload](#)
- struct [CFE_SB_SubEntries](#)
SB Previous Subscriptions Entry.
- struct [CFE_SB_AllSubscriptionsTlm_Payload](#)

Typedefs

- typedef struct [CFE_SB_WriteFileInfoCmd_Payload](#) [CFE_SB_WriteFileInfoCmd_Payload_t](#)
Write File Info Command Payload.
- typedef struct [CFE_SB_RouteCmd_Payload](#) [CFE_SB_RouteCmd_Payload_t](#)
Enable/Disable Route Command Payload.
- typedef struct [CFE_SB_HousekeepingTlm_Payload](#) [CFE_SB_HousekeepingTlm_Payload_t](#)
- typedef struct [CFE_SB_PipeDepthStats](#) [CFE_SB_PipeDepthStats_t](#)
SB Pipe Depth Statistics.
- typedef struct [CFE_SB_PipeInfoEntry](#) [CFE_SB_PipeInfoEntry_t](#)
SB Pipe Information File Entry.

- `typedef struct CFE_SB_StatsTlm_Payload CFE_SB_StatsTlm_Payload_t`
- `typedef struct CFE_SB_RoutingFileEntry CFE_SB_RoutingFileEntry_t`
SB Routing File Entry.
- `typedef struct CFE_SB_MsgMapFileEntry CFE_SB_MsgMapFileEntry_t`
SB Map File Entry.
- `typedef struct CFE_SB_SingleSubscriptionTlm_Payload CFE_SB_SingleSubscriptionTlm_Payload_t`
- `typedef struct CFE_SB_SubEntries CFE_SB_SubEntries_t`
SB Previous Subscriptions Entry.
- `typedef struct CFE_SB_AllSubscriptionsTlm_Payload CFE_SB_AllSubscriptionsTlm_Payload_t`

12.138.1 Detailed Description

Specification for the CFE Event Services (CFE_SB) command and telemetry message constant definitions.
For CFE_SB this is only the function/command code definitions

12.138.2 Typedef Documentation

12.138.2.1 `CFE_SB_AllSubscriptionsTlm_Payload_t` `typedef struct CFE_SB_AllSubscriptionsTlm_Payload CFE_SB_AllSubscriptionsTlm_Payload_t`

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.

12.138.2.2 `CFE_SB_HousekeepingTlm_Payload_t` `typedef struct CFE_SB_HousekeepingTlm_Payload CFE_SB_HousekeepingTlm_Payload_t`

Name Software Bus task housekeeping Packet

12.138.2.3 `CFE_SB_MsgMapFileEntry_t` `typedef struct CFE_SB_MsgMapFileEntry CFE_SB_MsgMapFileEntry_t`

SB Map File Entry.

Structure of one element of the map information in response to `CFE_SB_WRITE_MAP_INFO_CC`

12.138.2.4 `CFE_SB_PipeDepthStats_t` `typedef struct CFE_SB_PipeDepthStats CFE_SB_PipeDepthStats_t`

SB Pipe Depth Statistics.

Used in SB Statistics Telemetry Packet `CFE_SB_StatsTlm_t`

12.138.2.5 `CFE_SB_PipeInfoEntry_t` `typedef struct CFE_SB_PipeInfoEntry CFE_SB_PipeInfoEntry_t`

SB Pipe Information File Entry.

This statistics structure is output as part of the CFE SB "Send Pipe Info" command (CFE_SB_SEND_PIPE_INFO_CC). Previous versions of CFE simply wrote the internal CFE_SB_PipeD_t object to the file, but this also contains information such as pointers which are not relevant outside the running CFE process.

By defining the pipe info structure separately, it also provides some independence, such that the internal CFE_SB_PipeD_t definition can evolve without changing the binary format of the information file.

12.138.2.6 CFE_SB_RouteCmd_Payload_t `typedef struct CFE_SB_RouteCmd_Payload CFE_SB_RouteCmd_Payload_t`
Enable/Disable Route Command Payload.

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.

12.138.2.7 CFE_SB_RoutingFileEntry_t `typedef struct CFE_SB_RoutingFileEntry CFE_SB_RoutingFileEntry_t`
SB Routing File Entry.

Structure of one element of the routing information in response to [CFE_SB_WRITE_ROUTING_INFO_CC](#)

12.138.2.8 CFE_SB_SingleSubscriptionTlm_Payload_t `typedef struct CFE_SB_SingleSubscriptionTlm_Payload CFE_SB_SingleSubscriptionTlm_Payload_t`

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](#)

12.138.2.9 CFE_SB_StatsTlm_Payload_t `typedef struct CFE_SB_StatsTlm_Payload CFE_SB_StatsTlm_Payload_t`

Name SB Statistics Telemetry Packet

SB Statistics packet sent in response to [CFE_SB_SEND_SB_STATS_CC](#)

12.138.2.10 CFE_SB_SubEntries_t `typedef struct CFE_SB_SubEntries CFE_SB_SubEntries_t`
SB Previous Subscriptions Entry.

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](#)

12.138.2.11 CFE_SB_WriteFileInfoCmd_Payload_t `typedef struct CFE_SB_WriteFileInfoCmd_Payload CFE_SB_WriteFileInfoCmd_Payload_t`

Write File Info Command Payload.

This structure contains a generic definition used by SB commands that write to a file

12.139 cfe/modules/sb/config/default_cfe_sb_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_sb_topicids.h"
```

Macros

- `#define CFE_SB_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_CMD_TOPICID)`
/* 0x1803 */
- `#define CFE_SB_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_SEND_HK_TOPICID)`
/* 0x180B */

- #define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID)
/* 0x180E */
- #define CFE_SB_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_HK_TLM_TOPICID)
/* 0x0803 */
- #define CFE_SB_STATS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_STATS_TLM_TOPICID)
/* 0x080A */
- #define CFE_SB_ALLSUBS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_ALLSUBS_TLM_TOPICID)
/* 0x080D */
- #define CFE_SB_ONESUB_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_ONESUB_TLM_TOPICID)
/* 0x080E */

12.139.1 Detailed Description

CFE Event Services (CFE_SB) Application Message IDs

12.139.2 Macro Definition Documentation

12.139.2.1 CFE_SB_ALLSUBS_TLM_MID #define CFE_SB_ALLSUBS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_ALLSUBS_TLM_TOPICID)
/* 0x080D */
Definition at line 41 of file default_cfe_sb_msgids.h.

12.139.2.2 CFE_SB_CMD_MID #define CFE_SB_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_CMD_TOPICID)
/* 0x1803 */
Definition at line 32 of file default_cfe_sb_msgids.h.

12.139.2.3 CFE_SB_HK_TLM_MID #define CFE_SB_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_HK_TLM_TOPICID)
/* 0x0803 */
Definition at line 39 of file default_cfe_sb_msgids.h.

12.139.2.4 CFE_SB_ONESUB_TLM_MID #define CFE_SB_ONESUB_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_ONESUB_TLM_TOPICID)
/* 0x080E */
Definition at line 42 of file default_cfe_sb_msgids.h.

12.139.2.5 CFE_SB_SEND_HK_MID #define CFE_SB_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_SEND_HK_TOPICID)
/* 0x180B */
Definition at line 33 of file default_cfe_sb_msgids.h.

12.139.2.6 CFE_SB_STATS_TLM_MID #define CFE_SB_STATS_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_SB_STATS_TLM_TOPICID)
/* 0x080A */
Definition at line 40 of file default_cfe_sb_msgids.h.

12.139.2.7 CFE_SB_SUB_RPT_CTRL_MID #define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID)
/* 0x180E */
Definition at line 34 of file default_cfe_sb_msgids.h.

12.140 cfe/modules/sb/config/default_cfe_sb_msgstruct.h File Reference

```
#include "cfe_sb_msgdefs.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- struct [CFE_SB_NoopCmd](#)
- struct [CFE_SB_ResetCountersCmd](#)
- struct [CFE_SB_EnableSubReportingCmd](#)
- struct [CFE_SB_DisableSubReportingCmd](#)
- struct [CFE_SB_SendSbStatsCmd](#)
- struct [CFE_SB_SendPrevSubsCmd](#)
- struct [CFE_SB_SendHkCmd](#)
- struct [CFE_SB_WriteRoutingInfoCmd](#)
- struct [CFE_SB_WritePipeInfoCmd](#)
- struct [CFE_SB_WriteMapInfoCmd](#)
- struct [CFE_SB_EnableRouteCmd](#)
- struct [CFE_SB_DisableRouteCmd](#)
- struct [CFE_SB_HousekeepingTlm](#)
- struct [CFE_SB_StatsTlm](#)
- struct [CFE_SB_SingleSubscriptionTlm](#)
- struct [CFE_SB_AllSubscriptionsTlm](#)

Typedefs

- typedef struct [CFE_SB_NoopCmd](#) [CFE_SB_NoopCmd_t](#)
- typedef struct [CFE_SB_ResetCountersCmd](#) [CFE_SB_ResetCountersCmd_t](#)
- typedef struct [CFE_SB_EnableSubReportingCmd](#) [CFE_SB_EnableSubReportingCmd_t](#)
- typedef struct [CFE_SB_DisableSubReportingCmd](#) [CFE_SB_DisableSubReportingCmd_t](#)
- typedef struct [CFE_SB_SendSbStatsCmd](#) [CFE_SB_SendSbStatsCmd_t](#)
- typedef struct [CFE_SB_SendPrevSubsCmd](#) [CFE_SB_SendPrevSubsCmd_t](#)
- typedef struct [CFE_SB_SendHkCmd](#) [CFE_SB_SendHkCmd_t](#)
- typedef struct [CFE_SB_WriteRoutingInfoCmd](#) [CFE_SB_WriteRoutingInfoCmd_t](#)
- typedef struct [CFE_SB_WritePipeInfoCmd](#) [CFE_SB_WritePipeInfoCmd_t](#)
- typedef struct [CFE_SB_WriteMapInfoCmd](#) [CFE_SB_WriteMapInfoCmd_t](#)
- typedef struct [CFE_SB_EnableRouteCmd](#) [CFE_SB_EnableRouteCmd_t](#)
- typedef struct [CFE_SB_DisableRouteCmd](#) [CFE_SB_DisableRouteCmd_t](#)
- typedef struct [CFE_SB_HousekeepingTlm](#) [CFE_SB_HousekeepingTlm_t](#)
- typedef struct [CFE_SB_StatsTlm](#) [CFE_SB_StatsTlm_t](#)
- typedef struct [CFE_SB_SingleSubscriptionTlm](#) [CFE_SB_SingleSubscriptionTlm_t](#)
- typedef struct [CFE_SB_AllSubscriptionsTlm](#) [CFE_SB_AllSubscriptionsTlm_t](#)

12.140.1 Detailed Description

Purpose: cFE Executive Services (SB) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

12.140.2 Typedef Documentation

12.140.2.1 **CFE_SB_AllSubscriptionsTlm_t** `typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t`

12.140.2.2 **CFE_SB_DisableRouteCmd_t** `typedef struct CFE_SB_DisableRouteCmd CFE_SB_DisableRouteCmd_t`

12.140.2.3 **CFE_SB_DisableSubReportingCmd_t** `typedef struct CFE_SB_DisableSubReportingCmd CFE_SB_DisableSubReportin`

12.140.2.4 **CFE_SB_EnableRouteCmd_t** `typedef struct CFE_SB_EnableRouteCmd CFE_SB_EnableRouteCmd_t`

12.140.2.5 **CFE_SB_EnableSubReportingCmd_t** `typedef struct CFE_SB_EnableSubReportingCmd CFE_SB_EnableSubReportin`

12.140.2.6 **CFE_SB_HousekeepingTlm_t** `typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t`

12.140.2.7 **CFE_SB_NoopCmd_t** `typedef struct CFE_SB_NoopCmd CFE_SB_NoopCmd_t`

12.140.2.8 **CFE_SB_ResetCountersCmd_t** `typedef struct CFE_SB_ResetCountersCmd CFE_SB_ResetCountersCmd_t`

12.140.2.9 **CFE_SB_SendHkCmd_t** `typedef struct CFE_SB_SendHkCmd CFE_SB_SendHkCmd_t`

12.140.2.10 **CFE_SB_SendPrevSubsCmd_t** `typedef struct CFE_SB_SendPrevSubsCmd CFE_SB_SendPrevSubsCmd_t`

12.140.2.11 **CFE_SB_SendSbStatsCmd_t** `typedef struct CFE_SB_SendSbStatsCmd CFE_SB_SendSbStatsCmd_t`

12.140.2.12 **CFE_SB_SingleSubscriptionTlm_t** `typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscription`

12.140.2.13 **CFE_SB_StatsTlm_t** `typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t`

12.140.2.14 **CFE_SB_WriteMapInfoCmd_t** `typedef struct CFE_SB_WriteMapInfoCmd CFE_SB_WriteMapInfoCmd_t`

12.140.2.15 **CFE_SB_WritePipeInfoCmd_t** `typedef struct CFE_SB_WritePipeInfoCmd CFE_SB_WritePipeInfoCmd_t`

12.140.2.16 **CFE_SB_WriteRoutingInfoCmd_t** `typedef struct CFE_SB_WriteRoutingInfoCmd CFE_SB_WriteRoutingInfoCmd_t`

12.141 cfe/modules/sb/config/default_cfe_sb_platform_cfg.h File Reference

```
#include "cfe_sb_mission_cfg.h"
#include "cfe_sb_internal_cfg.h"
```

12.141.1 Detailed Description

CFE Software Bus (CFE_SB) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.142 cfe/modules/sb/config/default_cfe_sb_topicids.h File Reference

Macros

- #define CFE_MISSION_SB_CMD_TOPICID 3
- #define CFE_MISSION_SB_SEND_HK_TOPICID 11
- #define CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID 14
- #define CFE_MISSION_SB_HK_TLM_TOPICID 3
- #define CFE_MISSION_SB_STATS_TLM_TOPICID 10
- #define CFE_MISSION_SB_ALLSUBS_TLM_TOPICID 13
- #define CFE_MISSION_SB_ONESUB_TLM_TOPICID 14

12.142.1 Detailed Description

CFE Software Bus (CFE_SB) Application Topic IDs

12.142.2 Macro Definition Documentation

12.142.2.1 CFE_MISSION_SB_ALLSUBS_TLM_TOPICID #define CFE_MISSION_SB_ALLSUBS_TLM_TOPICID 13
Definition at line 50 of file default_cfe_sb_topicids.h.

12.142.2.2 CFE_MISSION_SB_CMD_TOPICID #define CFE_MISSION_SB_CMD_TOPICID 3

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE command messages

Limits

Not Applicable

Definition at line 35 of file default_cfe_sb_topicids.h.

12.142.2.3 CFE_MISSION_SB_HK_TLM_TOPICID #define CFE_MISSION_SB_HK_TLM_TOPICID 3

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE telemetry messages

Limits

Not Applicable

Definition at line 48 of file default_cfe_sb_topicids.h.

12.142.2.4 CFE_MISSION_SB_ONESUB_TLM_TOPICID #define CFE_MISSION_SB_ONESUB_TLM_TOPICID 14

Definition at line 51 of file default_cfe_sb_topicids.h.

12.142.2.5 CFE_MISSION_SB_SEND_HK_TOPICID #define CFE_MISSION_SB_SEND_HK_TOPICID 11

Definition at line 36 of file default_cfe_sb_topicids.h.

12.142.2.6 CFE_MISSION_SB_STATS_TLM_TOPICID #define CFE_MISSION_SB_STATS_TLM_TOPICID 10

Definition at line 49 of file default_cfe_sb_topicids.h.

12.142.2.7 CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID #define CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID 14

Definition at line 37 of file default_cfe_sb_topicids.h.

12.143 cfe/modules/sb/fsw/inc/cfe_sb_eventids.h File Reference

Macros

SB event IDs

- #define **CFE_SB_INIT_EID** 1
SB Initialization Event ID.
- #define **CFE_SB_CR_PIPE_BAD_ARG_EID** 2
SB Create Pipe API Bad Argument Event ID.
- #define **CFE_SB_MAX_PIPES_MET_EID** 3
SB Create Pipe API Max Pipes Exceeded Event ID.
- #define **CFE_SB_CR_PIPE_ERR_EID** 4
SB Create Pipe API Queue Create Failure Event ID.
- #define **CFE_SB_PIPE_ADDED_EID** 5
SB Create Pipe API Success Event ID.
- #define **CFE_SB_SUB_ARG_ERR_EID** 6
SB Subscribe API Bad Argument Event ID.
- #define **CFE_SB_DUP_SUBSCRIP_EID** 7
SB Subscribe API Duplicate MsgId Subscription Event ID.
- #define **CFE_SB_MAX_MSGS_MET_EID** 8
SB Subscribe API Max Subscriptions Exceeded Event ID.
- #define **CFE_SB_MAX_DESTS_MET_EID** 9

- #define **CFE_SB_SUBSCRIPTION_RCVD_EID** 10
 SB Subscribe API Success Event ID.
- #define **CFE_SB_UNSUB_ARG_ERR_EID** 11
 SB Unsubscribe API Bad Argument Event ID.
- #define **CFE_SB_UNSUB_NO_SUBS_EID** 12
 SB Unsubscribe API No MsgId Subscription Event ID.
- #define **CFE_SB_SEND_BAD_ARG_EID** 13
 SB Transmit API Bad Argument Event ID.
- #define **CFE_SB_SEND_NO_SUBS_EID** 14
 SB Transmit API No MsgId Subscribers Event ID.
- #define **CFE_SB_MSG_TOO_BIG_EID** 15
 SB Transmit API Message Size Limit Exceeded Event ID.
- #define **CFE_SB_GET_BUF_ERR_EID** 16
 SB Transmit API Buffer Request Failure Event ID.
- #define **CFE_SB_MSGID_LIM_ERR_EID** 17
 SB Transmit API MsgId Pipe Limit Exceeded Event ID.
- #define **CFE_SB_RCV_BAD_ARG_EID** 18
 SB Receive Buffer API Bad Argument Event ID.
- #define **CFE_SB_BAD_PIPEID_EID** 19
 SB Receive Buffer API Invalid Pipe Event ID.
- #define **CFE_SB_DEST_BLK_ERR_EID** 20
 SB Subscribe API Get Destination Block Failure Event ID.
- #define **CFE_SB_SEND_INV_MSGID_EID** 21
 SB Transmit API Invalid MsgId Event ID.
- #define **CFE_SB_SUBSCRIPTION_RPT_EID** 22
 SB Subscription Report Sent Event ID.
- #define **CFE_SB_HASHCOLLISION_EID** 23
 SB Subscribe API Message Table Hash Collision Event ID.
- #define **CFE_SB_Q_FULL_ERR_EID** 25
 SB Transmit API Pipe Overflow Event ID.
- #define **CFE_SB_Q_WR_ERR_EID** 26
 SB Transmit API Queue Write Failure Event ID.
- #define **CFE_SB_Q_RD_ERR_EID** 27
 SB Transmit API Queue Read Failure Event ID.
- #define **CFE_SB_CMD0_RCVD_EID** 28
 SB No-op Command Success Event ID.
- #define **CFE_SB_CMD1_RCVD_EID** 29
 SB Reset Counters Command Success Event ID.
- #define **CFE_SB SND_STATS_EID** 32
 SB Send Statistics Command Success Event ID.
- #define **CFE_SB_ENBL RTE1_EID** 33
 SB Enable Route Command Invalid MsgId/PipeID Pair Event ID.
- #define **CFE_SB_ENBL RTE2_EID** 34
 SB Enable Route Command Success Event ID.
- #define **CFE_SB_ENBL RTE3_EID** 35
 SB Enable Route Command Invalid MsgId or Pipe Event ID.
- #define **CFE_SB_DSBL RTE1_EID** 36
 SB Disable Route Command Invalid MsgId/PipeID Pair Event ID.
- #define **CFE_SB_DSBL RTE2_EID** 37
 SB Disable Route Command Success Event ID.
- #define **CFE_SB_DSBL RTE3_EID** 38
 SB Disable Route Command Invalid MsgId or Pipe Event ID.
- #define **CFE_SB SND RTG_EID** 39
 SB File Write Success Event ID.

- #define `CFE_SB SND RTG ERR1 EID` 40
SB File Write Create File Failure Event ID.
- #define `CFE_SB BAD CMD CODE EID` 42
SB Invalid Command Code Received Event ID.
- #define `CFE_SB BAD MSGID EID` 43
SB Invalid Message ID Received Event ID.
- #define `CFE_SB FULL SUB PKT EID` 44
SB Send Previous Subscriptions Command Full Packet Sent Event ID.
- #define `CFE_SB PART SUB PKT EID` 45
SB Send Previous Subscriptions Command Partial Packet Sent Event ID.
- #define `CFE_SB DEL PIPE ERR1 EID` 46
SB Pipe Delete API Bad Argument Event ID.
- #define `CFE_SB PIPE DELETED EID` 47
SB Pipe Delete API Success Event ID.
- #define `CFE_SB SUBSCRIPTION REMOVED EID` 48
SB Unsubscribe API Success Event ID.
- #define `CFE_SB FILEWRITE ERR EID` 49
SB File Write Failed Event ID.
- #define `CFE_SB SUB INV PIPE EID` 50
SB Subscribe API Invalid Pipe Event ID.
- #define `CFE_SB SUB INV CALLER EID` 51
SB Subscribe API Not Owner Event ID.
- #define `CFE_SB UNSUB INV PIPE EID` 52
SB Unsubscribe API Invalid Pipe Event ID.
- #define `CFE_SB UNSUB INV CALLER EID` 53
SB Unsubscribe API Not Owner Event ID.
- #define `CFE_SB DEL PIPE ERR2 EID` 54
SB Delete Pipe API Not Owner Event ID.
- #define `CFE_SB SETPIPEOPTS ID ERR EID` 55
SB Set Pipe Opts API Invalid Pipe Event ID.
- #define `CFE_SB SETPIPEOPTS OWNER ERR EID` 56
SB Set Pipe Opts API Not Owner Event ID.
- #define `CFE_SB SETPIPEOPTS EID` 57
SB Set Pipe Opts API Success Event ID.
- #define `CFE_SB GETPIPEOPTS ID ERR EID` 58
SB Get Pipe Opts API Invalid Pipe Event ID.
- #define `CFE_SB GETPIPEOPTS PTR ERR EID` 59
SB Get Pipe Opts API Invalid Pointer Event ID.
- #define `CFE_SB GETPIPEOPTS EID` 60
SB Get Pipe Opts API Success Event ID.
- #define `CFE_SB GETPIPE NAME EID` 62
SB Get Pipe Name API Success Event ID.
- #define `CFE_SB GETPIPE NAME NULL PTR EID` 63
SB Get Pipe Name API Invalid Pointer Event ID.
- #define `CFE_SB GETPIPE NAME ID ERR EID` 64
SB Get Pipe Name API Invalid Pipe or Resource Event ID.
- #define `CFE_SB GETPIPE ID BYNAME EID` 65
SB Get Pipe ID By Name API Success Event ID.
- #define `CFE_SB GETPIPE ID BYNAME NULL PTR EID` 66
SB Get Pipe ID By Name API Invalid Pointer Event ID.
- #define `CFE_SB GETPIPE ID BYNAME NAME ERR EID` 67
SB Get Pipe ID By Name API Name Not Found Or ID Not Matched Event ID.
- #define `CFE_SB LEN ERR EID` 68
SB Invalid Command Length Event ID.
- #define `CFE_SB CR PIPE NAME TAKEN EID` 69

- #define [CFE_SB_CR_PIPE_NO_FREE_EID](#) 70
SB Create Pipe API Name Taken Event ID.
- #define [CFE_SB_SEND_MESSAGE_INTEGRITY_FAIL_EID](#) 71
SB Create Pipe API Queues Exhausted Event ID.
- #define [CFE_SB_RCV_MESSAGE_INTEGRITY_FAIL_EID](#) 72
SB integrity actions on transmit message failure event.
- #define [CFE_SB_RCV_MESSAGE_INTEGRITY_FAIL_EID](#) 72
SB validation of received message failure event.

12.143.1 Detailed Description

cFE Software Bus Services Event IDs

12.143.2 Macro Definition Documentation

12.143.2.1 CFE_SB_BAD_CMD_CODE_EID #define CFE_SB_BAD_CMD_CODE_EID 42
SB Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE_SB_CMD_MID](#) or [CFE_SB_SUB_RPT_CTRL_MID](#) received on the SB message pipe. OVERLOADED

Definition at line 461 of file cfe_sb_eventids.h.

12.143.2.2 CFE_SB_BAD_MSGID_EID #define CFE_SB_BAD_MSGID_EID 43
SB Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the SB message pipe.

Definition at line 472 of file cfe_sb_eventids.h.

12.143.2.3 CFE_SB_BAD_PIPEID_EID #define CFE_SB_BAD_PIPEID_EID 19
SB Receive Buffer API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE_SB_ReceiveBuffer](#) API failure due to an invalid Pipe ID.

Definition at line 244 of file cfe_sb_eventids.h.

12.143.2.4 CFE_SB_CMD0_RCVD_EID #define CFE_SB_CMD0_RCVD_EID 28
SB No-op Command Success Event ID.

Type: INFORMATION

Cause:

[SB NO-OP Command](#) success.

Definition at line 335 of file cfe_sb_eventids.h.

12.143.2.5 CFE_SB_CMD1_RCVD_EID #define CFE_SB_CMD1_RCVD_EID 29
SB Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[SB Reset Counters Command](#) success.

Definition at line 346 of file cfe_sb_eventids.h.

12.143.2.6 CFE_SB_CR_PIPE_BAD_ARG_EID #define CFE_SB_CR_PIPE_BAD_ARG_EID 2
SB Create Pipe API Bad Argument Event ID.

Type: ERROR

Cause:

[CFE_SB_CreatePipe](#) API failure due to a bad input argument.

Definition at line 53 of file cfe_sb_eventids.h.

12.143.2.7 CFE_SB_CR_PIPE_ERR_EID #define CFE_SB_CR_PIPE_ERR_EID 4
SB Create Pipe API Queue Create Failure Event ID.

Type: ERROR

Cause:

[CFE_SB_CreatePipe](#) API failure creating the queue.

Definition at line 75 of file cfe_sb_eventids.h.

12.143.2.8 CFE_SB_CR_PIPE_NAME_TAKEN_EID #define CFE_SB_CR_PIPE_NAME_TAKEN_EID 69
SB Create Pipe API Name Taken Event ID.

Type: ERROR

Cause:

[CFE_SB_CreatePipe](#) API failure due to pipe name taken.
Definition at line 750 of file cfe_sb_eventids.h.

12.143.2.9 CFE_SB_CR_PIPE_NO_FREE_EID #define CFE_SB_CR_PIPE_NO_FREE_EID 70
SB Create Pipe API Queues Exhausted Event ID.

Type: ERROR

Cause:

[CFE_SB_CreatePipe](#) API failure due to no free queues.
Definition at line 761 of file cfe_sb_eventids.h.

12.143.2.10 CFE_SB_DEL_PIPE_ERR1_EID #define CFE_SB_DEL_PIPE_ERR1_EID 46
SB Pipe Delete API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Delete Pipe API failed due to an invalid input argument.
Definition at line 507 of file cfe_sb_eventids.h.

12.143.2.11 CFE_SB_DEL_PIPE_ERR2_EID #define CFE_SB_DEL_PIPE_ERR2_EID 54
SB Delete Pipe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Delete Pipe API failed due to not being the pipe owner.
Definition at line 595 of file cfe_sb_eventids.h.

12.143.2.12 CFE_SB_DEST_BLK_ERR_EID #define CFE_SB_DEST_BLK_ERR_EID 20
SB Subscribe API Get Destination Block Failure Event ID.

Type: ERROR

Cause:

An SB Subscribe API call failed to get a destination block.
Definition at line 255 of file cfe_sb_eventids.h.

12.143.2.13 CFE_SB_DSBL RTE1_EID #define CFE_SB_DSBL RTE1_EID 36
SB Disable Route Command Invalid MsgId/PipeId Pair Event ID.

Type: ERROR

Cause:

[SB Disable Route Command](#) failure due to the Message ID not being subscribed to the pipe.
Definition at line 404 of file cfe_sb_eventids.h.

12.143.2.14 CFE_SB_DSBL RTE2_EID #define CFE_SB_DSBL RTE2_EID 37
SB Disable Route Command Success Event ID.

Type: DEBUG

Cause:

[SB Disable Route Command](#) success.
Definition at line 415 of file cfe_sb_eventids.h.

12.143.2.15 CFE_SB_DSBL RTE3_EID #define CFE_SB_DSBL RTE3_EID 38
SB Disable Route Command Invalid MsgId or Pipe Event ID.

Type: ERROR

Cause:

[SB Disable Route Command](#) failure due to an invalid MsgId or Pipe.
Definition at line 427 of file cfe_sb_eventids.h.

12.143.2.16 CFE_SB_DUP_SUBSCRIP_EID #define CFE_SB_DUP_SUBSCRIP_EID 7
SB Subscribe API Duplicate MsgId Subscription Event ID.

Type: INFORMATION

Cause:

An SB Subscribe API was called with a Message ID that was already subscribed on the pipe on the pipe.
Definition at line 109 of file cfe_sb_eventids.h.

12.143.2.17 CFE_SB_ENBL RTE1_EID #define CFE_SB_ENBL RTE1_EID 33
SB Enable Route Command Invalid MsgId/PipeID Pair Event ID.

Type: ERROR

Cause:

[SB Enable Route Command](#) failure due to the Message ID not being subscribed to the pipe.
Definition at line 369 of file cfe_sb_eventids.h.

12.143.2.18 CFE_SB_ENBL RTE2_EID #define CFE_SB_ENBL RTE2_EID 34
SB Enable Route Command Success Event ID.

Type: DEBUG

Cause:

[SB Enable Route Command](#) success.
Definition at line 380 of file cfe_sb_eventids.h.

12.143.2.19 CFE_SB_ENBL RTE3_EID #define CFE_SB_ENBL RTE3_EID 35
SB Enable Route Command Invalid MsgId or Pipe Event ID.

Type: ERROR

Cause:

[SB Enable Route Command](#) failure due to an invalid MsgId or Pipe.
Definition at line 392 of file cfe_sb_eventids.h.

12.143.2.20 CFE_SB_FILEWRITE_ERR_EID #define CFE_SB_FILEWRITE_ERR_EID 49
SB File Write Failed Event ID.

Type: ERROR

Cause:

An SB file write failure encountered when writing to the file.
Definition at line 540 of file cfe_sb_eventids.h.

12.143.2.21 CFE_SB_FULL_SUB_PKT_EID #define CFE_SB_FULL_SUB_PKT_EID 44
SB Send Previous Subscriptions Command Full Packet Sent Event ID.

Type: DEBUG

Cause:

[SB Send Previous Subscriptions Command](#) processing sent a full subscription packet.
Definition at line 484 of file cfe_sb_eventids.h.

12.143.2.22 CFE_SB_GET_BUF_ERR_EID #define CFE_SB_GET_BUF_ERR_EID 16
SB Transmit API Buffer Request Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API call buffer request failed.
Definition at line 210 of file cfe_sb_eventids.h.

12.143.2.23 CFE_SB_GETPIPEIDBYNAME_EID #define CFE_SB_GETPIPEIDBYNAME_EID 65
SB Get Pipe ID By Name API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_GetPipeIdByName](#) success.
Definition at line 705 of file cfe_sb_eventids.h.

12.143.2.24 CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID #define CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID 67

SB Get Pipe ID By Name API Name Not Found Or ID Not Matched Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeIdByName](#) failure due to name not found or ID mismatch. OVERLOADED
Definition at line 727 of file cfe_sb_eventids.h.

12.143.2.25 CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID #define CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID 66

SB Get Pipe ID By Name API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeIdByName](#) failure due to invalid pointer.
Definition at line 716 of file cfe_sb_eventids.h.

12.143.2.26 CFE_SB_GETPIPENAME_EID #define CFE_SB_GETPIPENAME_EID 62

SB Get Pipe Name API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_GetPipeName](#) success.
Definition at line 672 of file cfe_sb_eventids.h.

12.143.2.27 CFE_SB_GETPIPENAME_ID_ERR_EID #define CFE_SB_GETPIPENAME_ID_ERR_EID 64

SB Get Pipe Name API Invalid Pipe or Resource Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeName](#) failure due to invalid pipe ID or failure in retrieving resource name. OVERLOADED
Definition at line 694 of file cfe_sb_eventids.h.

12.143.2.28 CFE_SB_GETPIPENAME_NULL_PTR_EID #define CFE_SB_GETPIPENAME_NULL_PTR_EID 63
SB Get Pipe Name API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeName](#) failure due to invalid pointer.
Definition at line 683 of file cfe_sb_eventids.h.

12.143.2.29 CFE_SB_GETPIPEOPTS_EID #define CFE_SB_GETPIPEOPTS_EID 60
SB Get Pipe Opt API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_GetPipeOpt](#) success.
Definition at line 661 of file cfe_sb_eventids.h.

12.143.2.30 CFE_SB_GETPIPEOPTS_ID_ERR_EID #define CFE_SB_GETPIPEOPTS_ID_ERR_EID 58
SB Get Pipe Opt API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeOpt](#) failure due to invalid pipe ID.
Definition at line 639 of file cfe_sb_eventids.h.

12.143.2.31 CFE_SB_GETPIPEOPTS_PTR_ERR_EID #define CFE_SB_GETPIPEOPTS_PTR_ERR_EID 59
SB Get Pipe Opt API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE_SB_GetPipeOpt](#) failure due to invalid pointer.
Definition at line 650 of file cfe_sb_eventids.h.

12.143.2.32 CFE_SB_HASHCOLLISION_EID #define CFE_SB_HASHCOLLISION_EID 23
SB Subscribe API Message Table Hash Collision Event ID.

Type: DEBUG

Cause:

An SB Subscribe API call caused a message table hash collision, which will impact message transmission performance. This can be resolved by deconflicting MsgId values or increasing [CFE_PLATFORM_SB_MAX_MSG_IDS](#).
Definition at line 290 of file cfe_sb_eventids.h.

12.143.2.33 CFE_SB_INIT_EID #define CFE_SB_INIT_EID 1
SB Initialization Event ID.

Type: INFORMATION

Cause:

Software Bus Services Task initialization complete.
Definition at line 42 of file cfe_sb_eventids.h.

12.143.2.34 CFE_SB_LEN_ERR_EID #define CFE_SB_LEN_ERR_EID 68
SB Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE_SB_CMD_MID](#) or [CFE_SB_SUB_RPT_CTRL_MID](#) received on the SB message pipe.
Definition at line 739 of file cfe_sb_eventids.h.

12.143.2.35 CFE_SB_MAX_DESTS_MET_EID #define CFE_SB_MAX_DESTS_MET_EID 9
SB Subscribe API Max Destinations Exceeded Event ID.

Type: ERROR

Cause:

An SB Subscribe API was called with a message id that already has the maximum allowed number of destinations.
Definition at line 133 of file cfe_sb_eventids.h.

12.143.2.36 CFE_SB_MAX_MSGS_MET_EID #define CFE_SB_MAX_MSGS_MET_EID 8
SB Subscribe API Max Subscriptions Exceeded Event ID.

Type: ERROR

Cause:

An SB Subscribe API was called on a pipe that already has the maximum allowed number of subscriptions.
Definition at line 121 of file cfe_sb_eventids.h.

12.143.2.37 CFE_SB_MAX_PIPES_MET_EID #define CFE_SB_MAX_PIPES_MET_EID 3
SB Create Pipe API Max Pipes Exceeded Event ID.

Type: ERROR

Cause:

[CFE_SB_CreatePipe](#) API failure to do maximum number of pipes being exceeded.
Definition at line 64 of file cfe_sb_eventids.h.

12.143.2.38 CFE_SB_MSG_TOO_BIG_EID #define CFE_SB_MSG_TOO_BIG_EID 15
SB Transmit API Message Size Limit Exceeded Event ID.

Type: ERROR

Cause:

An SB Transmit API was called with a message that is too big.
Definition at line 199 of file cfe_sb_eventids.h.

12.143.2.39 CFE_SB_MSGID_LIM_ERR_EID #define CFE_SB_MSGID_LIM_ERR_EID 17
SB Transmit API MsgId Pipe Limit Exceeded Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed to deliver the MsgId to a pipe due to the limit for the number of messages with that MsgId for that pipe being exceeded.
Definition at line 222 of file cfe_sb_eventids.h.

12.143.2.40 CFE_SB_PART_SUB_PKT_EID #define CFE_SB_PART_SUB_PKT_EID 45
SB Send Previous Subscriptions Command Partial Packet Sent Event ID.

Type: DEBUG

Cause:

[SB Send Previous Subscriptions Command](#) processing sent a partial subscription packet.
Definition at line 496 of file cfe_sb_eventids.h.

12.143.2.41 CFE_SB_PIPE_ADDED_EID #define CFE_SB_PIPE_ADDED_EID 5
SB Create Pipe API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_CreatePipe](#) API successfully completed.
Definition at line 86 of file cfe_sb_eventids.h.

12.143.2.42 CFE_SB_PIPE_DELETED_EID #define CFE_SB_PIPE_DELETED_EID 47
SB Pipe Delete API Success Event ID.

Type: DEBUG

Cause:

An SB Delete Pipe API successfully completed.
Definition at line 518 of file cfe_sb_eventids.h.

12.143.2.43 CFE_SB_Q_FULL_ERR_EID #define CFE_SB_Q_FULL_ERR_EID 25
SB Transmit API Pipe Overflow Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed to deliver the Message ID to a pipe due to the pipe queue being full.
Definition at line 302 of file cfe_sb_eventids.h.

12.143.2.44 CFE_SB_Q_RD_ERR_EID #define CFE_SB_Q_RD_ERR_EID 27
SB Transmit API Queue Read Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API called failed due to a pipe queue read failure.
Definition at line 324 of file cfe_sb_eventids.h.

12.143.2.45 CFE_SB_Q_WR_ERR_EID #define CFE_SB_Q_WR_ERR_EID 26
SB Transmit API Queue Write Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed due to a pipe queue write failure.
Definition at line 313 of file cfe_sb_eventids.h.

12.143.2.46 CFE_SB_RCV_BAD_ARG_EID #define CFE_SB_RCV_BAD_ARG_EID 18
SB Receive Buffer API Bad Argument Event ID.

Type: ERROR

Cause:

[CFE_SB_ReceiveBuffer](#) API failure due to a bad input argument.
Definition at line 233 of file cfe_sb_eventids.h.

12.143.2.47 CFE_SB_RCV_MESSAGE_INTEGRITY_FAIL_EID #define CFE_SB_RCV_MESSAGE_INTEGRITY_FAI←
L_EID 72
SB validation of received message failure event.

Type: ERROR

Cause:

A CFE SB receive transaction has rejected a message due to failure of the associated message integrity action(s).
Definition at line 785 of file cfe_sb_eventids.h.

12.143.2.48 CFE_SB_SEND_BAD_ARG_EID #define CFE_SB_SEND_BAD_ARG_EID 13
SB Transmit API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Transmit API failed due to an invalid input argument.
Definition at line 177 of file cfe_sb_eventids.h.

12.143.2.49 CFE_SB_SEND_INV_MSGID_EID #define CFE_SB_SEND_INV_MSGID_EID 21
SB Transmit API Invalid MsgId Event ID.

Type: ERROR

Cause:

An SB Transmit API was called with an invalid message ID.
Definition at line 266 of file cfe_sb_eventids.h.

12.143.2.50 CFE_SB_SEND_MESSAGE_INTEGRITY_FAIL_EID #define CFE_SB_SEND_MESSAGE_INTEGRITY_FAIL_EID 71
SB integrity actions on transmit message failure event.

Type: ERROR

Cause:

A CFE SB transmit transaction has rejected a message due to failure of the associated message integrity action(s).
Definition at line 773 of file cfe_sb_eventids.h.

12.143.2.51 CFE_SB_SEND_NO_SUBS_EID #define CFE_SB_SEND_NO_SUBS_EID 14
SB Transmit API No MsgId Subscribers Event ID.

Type: INFORMATION

Cause:

An SB Transmit API was called with a Message ID with no subscriptions.
Definition at line 188 of file cfe_sb_eventids.h.

12.143.2.52 CFE_SB_SETPipeOPTS_EID #define CFE_SB_SETPipeOPTS_EID 57
SB Set Pipe Opt API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_SetPipeOpt](#) success.

Definition at line 628 of file cfe_sb_eventids.h.

12.143.2.53 CFE_SB_SETPipeOPTS_ID_ERR_EID #define CFE_SB_SETPipeOPTS_ID_ERR_EID 55
SB Set Pipe Opt API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE_SB_SetPipeOpt](#) API failure due to an invalid pipe ID

Definition at line 606 of file cfe_sb_eventids.h.

12.143.2.54 CFE_SB_SETPipeOPTS_OWNER_ERR_EID #define CFE_SB_SETPipeOPTS_OWNER_ERR_EID 56
SB Set Pipe Opt API Not Owner Event ID.

Type: ERROR

Cause:

[CFE_SB_SetPipeOpt](#) API failure due to not being the pipe owner.

Definition at line 617 of file cfe_sb_eventids.h.

12.143.2.55 CFE_SB_SND_RTG_EID #define CFE_SB_SND_RTG_EID 39
SB File Write Success Event ID.

Type: DEBUG

Cause:

An SB file write successfully completed. OVERLOADED

Definition at line 438 of file cfe_sb_eventids.h.

12.143.2.56 CFE_SB SND RTG_ERR1_EID #define CFE_SB SND RTG_ERR1_EID 40
SB File Write Create File Failure Event ID.

Type: ERROR

Cause:

An SB file write failure due to file creation error. OVERLOADED
Definition at line 449 of file cfe_sb_eventids.h.

12.143.2.57 CFE_SB SND STATS_EID #define CFE_SB SND STATS_EID 32
SB Send Statistics Command Success Event ID.

Type: DEBUG

Cause:

[SB Send Statistics Command](#) success.
Definition at line 357 of file cfe_sb_eventids.h.

12.143.2.58 CFE_SB SUB ARG_ERR_EID #define CFE_SB SUB ARG_ERR_EID 6
SB Subscribe API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to an invalid input argument.
Definition at line 97 of file cfe_sb_eventids.h.

12.143.2.59 CFE_SB SUB INV_CALLER_EID #define CFE_SB SUB INV_CALLER_EID 51
SB Subscribe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to not being the pipe owner.
Definition at line 562 of file cfe_sb_eventids.h.

12.143.2.60 CFE_SB_SUB_INV_PIPE_EID #define CFE_SB_SUB_INV_PIPE_EID 50
SB Subscribe API Invalid Pipe Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to an invalid pipe ID.
Definition at line 551 of file cfe_sb_eventids.h.

12.143.2.61 CFE_SB_SUBSCRIPTION_RCVD_EID #define CFE_SB_SUBSCRIPTION_RCVD_EID 10
SB Subscribe API Success Event ID.

Type: DEBUG

Cause:

An SB Subscribe API completed successfully.
Definition at line 144 of file cfe_sb_eventids.h.

12.143.2.62 CFE_SB_SUBSCRIPTION_REMOVED_EID #define CFE_SB_SUBSCRIPTION_REMOVED_EID 48
SB Unsubscribe API Success Event ID.

Type: DEBUG

Cause:

An SB Unsubscribe API successfully completed.
Definition at line 529 of file cfe_sb_eventids.h.

12.143.2.63 CFE_SB_SUBSCRIPTION_RPT_EID #define CFE_SB_SUBSCRIPTION_RPT_EID 22
SB Subscription Report Sent Event ID.

Type: DEBUG

Cause:

SB Subscription Report sent in response to a successful subscription.
Definition at line 277 of file cfe_sb_eventids.h.

12.143.2.64 CFE_SB_UNSUB_ARG_ERR_EID #define CFE_SB_UNSUB_ARG_ERR_EID 11
SB Unsubscribe API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to an invalid input argument.

Definition at line 155 of file cfe_sb_eventids.h.

12.143.2.65 CFE_SB_UNSUB_INV_CALLER_EID #define CFE_SB_UNSUB_INV_CALLER_EID 53
SB Unsubscribe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to not being the pipe owner.

Definition at line 584 of file cfe_sb_eventids.h.

12.143.2.66 CFE_SB_UNSUB_INV_PIPE_EID #define CFE_SB_UNSUB_INV_PIPE_EID 52
SB Unsubscribe API Invalid Pipe Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to an invalid pipe ID.

Definition at line 573 of file cfe_sb_eventids.h.

12.143.2.67 CFE_SB_UNSUB_NO_SUBS_EID #define CFE_SB_UNSUB_NO_SUBS_EID 12
SB Unsubscribe API No MsgId Subscription Event ID.

Type: INFORMATION

Cause:

An SB Unsubscribe API was called with a Message ID that wasn't subscribed on the pipe

Definition at line 166 of file cfe_sb_eventids.h.

12.144 cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_mission_cfg.h"
```

Data Structures

- struct [CFE_TBL_File_Hdr](#)

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.

- typedef struct [CFE_TBL_File_Hdr](#) [CFE_TBL_File_Hdr_t](#)

The definition of the header fields that are included in CFE Table Data files.

Enumerations

- enum [CFE_TBL_BufferSelect](#) { [CFE_TBL_BufferSelect_INACTIVE](#) = 0, [CFE_TBL_BufferSelect_ACTIVE](#) = 1 }

Label definitions associated with CFE_TBL_BufferSelect_Enum_t.

12.144.1 Detailed Description

Declarations and prototypes for cfe_tbl_extern_typedefs module

12.144.2 Typedef Documentation

12.144.2.1 [CFE_TBL_BufferSelect_Enum_t](#) typedef uint16 [CFE_TBL_BufferSelect_Enum_t](#)

Selects the buffer to operate on for validate or dump commands.

See also

enum [CFE_TBL_BufferSelect](#)

Definition at line 53 of file default_cfe_tbl_extern_typedefs.h.

12.144.2.2 [CFE_TBL_File_Hdr_t](#) typedef struct [CFE_TBL_File_Hdr](#) [CFE_TBL_File_Hdr_t](#)

The definition of the header fields that are included in CFE Table Data files.

This header follows the CFE_FS header and precedes the actual table data.

Note

The Offset and NumBytes fields in the table header are to 32 bits for backward compatibility with existing CFE versions. This means that even on 64-bit CPUs, individual table files will be limited to 4GiB in size.

12.144.3 Enumeration Type Documentation

12.144.3.1 [CFE_TBL_BufferSelect](#) enum [CFE_TBL_BufferSelect](#)

Label definitions associated with CFE_TBL_BufferSelect_Enum_t.

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 35 of file default_cfe_tbl_extern_typedefs.h.

12.145 cfe/modules/tbl/config/default_cfe_tbl_fcncodes.h File Reference

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

12.145.1 Detailed Description

Specification for the CFE Event Services (CFE_TBL) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.145.2 Macro Definition Documentation

12.145.2.1 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_AbortLoadCmd_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 461 of file default_cfe_tbl_fcncodes.h.

12.145.2.2 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_ActivateCmd_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):

- The specified table name was not found in the table registry.
- The table was registered as a "dump only" type and thus cannot be activated
- The table buffer has not been validated.

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 299 of file default_cfe_tbl_fcncodes.h.

12.145.2.3 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_DeleteCDSCmd_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 422 of file default_cfe_tbl_fcncodes.h.

12.145.2.4 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_DumpCmd_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 202 of file default_cfe_tbl_fcncodes.h.

12.145.2.5 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_DumpRegistryCmd_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):

- A table registry dump is already in progress, not yet completed
- The specified DumpFilename could not be parsed
- 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 343 of file default_cfe_tbl_fcncodes.h.

12.145.2.6 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_LoadCmd_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 has an invalid or incorrect size. The size of the image file must match the size field within in the header, and must also match the expected size of the table indicated in the registry.
- 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.
- 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 159 of file default_cfe_tbl_fcncodes.h.

12.145.2.7 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_NoopCmd_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 68 of file default_cfe_tbl_fcncodes.h.

12.145.2.8 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)
- Number of completed table validations (\$sc_\$cpu_TBL_ValCompldCtr)

Command Mnemonic(s) \$sc_\$cpu_TBL_ResetCtrs

Command Structure

[CFE_TBL_ResetCountersCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_TBL_CMDPC` - command execution counter will be reset to 0
- 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 109 of file default_cfe_tbl_fcncodes.h.

12.145.2.9 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_SendRegistryCmd_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 378 of file default_cfe_tbl_fcncodes.h.

12.145.2.10 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_ValidateCmd_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 validated 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 259 of file default_cfe_tbl_fcncodes.h.

12.146 cfe/modules/tbl/config/default_cfe_tbl_interface_cfg.h File Reference

Macros

- `#define CFE_MISSION_TBL_MAX_NAME_LENGTH 16`
- `#define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)`

12.146.1 Detailed Description

CFE Table Services (CFE_TBL) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.146.2 Macro Definition Documentation

12.146.2.1 CFE_MISSION_TBL_MAX_FULL_NAME_LEN `#define CFE_MISSION_TBL_MAX_FULL_NAME_L←
EN (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 69 of file default_cfe_tbl_interface_cfg.h.

12.146.2.2 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 characters) of the table name ('TblName') portion of a Full Table Name of the following form: "ApplicationName.TblName"

This length does not need to include an extra character for NULL termination.

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 49 of file default_cfe_tbl_interface_cfg.h.

12.147 cfe/modules/tbl/config/default_cfe_tbl_internal_cfg.h File Reference**Macros**

- #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_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) ((uint32)(_C1) << 24 | (uint32)(_C2) << 16 | (uint32)(_C3) << 8 | (uint32)(_C4))
- #define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)
- #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 (1)
- #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

12.147.1 Detailed Description

CFE Table Services (CFE_TBL) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.147.2 Macro Definition Documentation

12.147.2.1 CFE_PLATFORM_TBL_BUF_MEMORY_BYTES #define CFE_PLATFORM_TBL_BUF_MEMORY_BYT←
ES 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 75 of file default_cfe_tbl_internal_cfg.h.

12.147.2.2 CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE #define CFE_PLATFORM_TBL_DEFAULT_REG_D←
UMP_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 189 of file default_cfe_tbl_internal_cfg.h.

12.147.2.3 CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES #define CFE_PLATFORM_TBL_MAX_CRITICAL_TA←
BLES 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_PLATFORM_E→S_CDS_MAX_NUM_ENTRIES.

Definition at line 130 of file default_cfe_tbl_internal_cfg.h.

12.147.2.4 CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE #define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SI→ZE 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_TBL_BUF_MEMORY_B](#)

Definition at line 87 of file default_cfe_tbl_internal_cfg.h.

12.147.2.5 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_PLATFORM_TBL_MAX_NUM_TABLES](#)) and should be set higher if tables are shared between applications.

Definition at line 143 of file default_cfe_tbl_internal_cfg.h.

12.147.2.6 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 116 of file default_cfe_tbl_internal_cfg.h.

12.147.2.7 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 176 of file default_cfe_tbl_internal_cfg.h.

12.147.2.8 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 158 of file default_cfe_tbl_internal_cfg.h.

12.147.2.9 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_LOADS](#) 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_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS](#) number of tables to fit into [CFE_PLATFORM_TBL_BUF_MEMORY_BYT](#)

Definition at line 103 of file default_cfe_tbl_internal_cfg.h.

12.147.2.10 CFE_PLATFORM_TBL_START_TASK_PRIORITY #define CFE_PLATFORM_TBL_START_TASK_PRIO←
RITY 70

Purpose Define TBL Task Priority

Description:

Defines the cFE_TBL Task priority.

Limits

Not Applicable

Definition at line 44 of file default_cfe_tbl_internal_cfg.h.

12.147.2.11 CFE_PLATFORM_TBL_START_TASK_STACK_SIZE #define CFE_PLATFORM_TBL_START_TASK_S←
TACK_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 parameter. There are no restrictions on the upper limit however, the maximum stack 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 59 of file default_cfe_tbl_internal_cfg.h.

12.147.2.12 CFE_PLATFORM_TBL_U32FROM4CHARS #define CFE_PLATFORM_TBL_U32FROM4CHARS (←
_C1,
_C2,
_C3,
_C4) ((uint32) (_C1) << 24 | (uint32) (_C2) << 16 | (uint32) (_C3) << 8 | (uint32) (←
_C4))

Definition at line 211 of file default_cfe_tbl_internal_cfg.h.

12.147.2.13 CFE_PLATFORM_TBL_VALID_PRID_1 #define CFE_PLATFORM_TBL_VALID_PRID_1 (1)

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 260 of file default_cfe_tbl_internal_cfg.h.

12.147.2.14 CFE_PLATFORM_TBL_VALID_PRID_2 #define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CH
'b', 'c', 'd'))

Definition at line 261 of file default_cfe_tbl_internal_cfg.h.

12.147.2.15 CFE_PLATFORM_TBL_VALID_PRID_3 #define CFE_PLATFORM_TBL_VALID_PRID_3 0

Definition at line 262 of file default_cfe_tbl_internal_cfg.h.

12.147.2.16 CFE_PLATFORM_TBL_VALID_PRID_4 #define CFE_PLATFORM_TBL_VALID_PRID_4 0

Definition at line 263 of file default_cfe_tbl_internal_cfg.h.

12.147.2.17 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 246 of file default_cfe_tbl_internal_cfg.h.

12.147.2.18 CFE_PLATFORM_TBL_VALID_SCID_1 #define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)

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 226 of file default_cfe_tbl_internal_cfg.h.

12.147.2.19 CFE_PLATFORM_TBL_VALID_SCID_2 #define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CH
'b', 'c', 'd'))

Definition at line 227 of file default_cfe_tbl_internal_cfg.h.

12.147.2.20 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 208 of file default_cfe_tbl_internal_cfg.h.

12.148 cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h File Reference

```
#include "cfe_tbl_interface_cfg.h"
```

12.148.1 Detailed Description

CFE Event Services (CFE_TBL) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.149 cfe/modules/tbl/config/default_cfe_tbl_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_tbl_fcncodes.h"
#include "cfe_tbl_msgdefs.h"
#include "cfe_tbl_msgstruct.h"
```

12.149.1 Detailed Description

Specification for the CFE Event Services (CFE_TBL) command and telemetry message data types.

This is a compatibility header for the "cfe_tbl_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.150 cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_time_extern_typedefs.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_tbl_fcncodes.h"
```

Data Structures

- struct [CFE_TBL_LoadCmd_Payload](#)
Load Table Command Payload.
- struct [CFE_TBL_DumpCmd_Payload](#)
Dump Table Command Payload.
- struct [CFE_TBL_ValidateCmd_Payload](#)
Validate Table Command Payload.
- struct [CFE_TBL_ActivateCmd_Payload](#)
Activate Table Command Payload.
- struct [CFE_TBL_DumpRegistryCmd_Payload](#)
Dump Registry Command Payload.
- struct [CFE_TBL_SendRegistryCmd_Payload](#)
Send Table Registry Command Payload.
- struct [CFE_TBL_DelCDSCmd_Payload](#)
Delete Critical Table CDS Command Payload.
- struct [CFE_TBL_AbortLoadCmd_Payload](#)
Abort Load Command Payload.
- struct [CFE_TBL_NotifyCmd_Payload](#)
Table Management Notification Command Payload.
- struct [CFE_TBL_HousekeepingTlm_Payload](#)
- struct [CFE_TBL_TblRegPacket_Payload](#)

Typedefs

- typedef struct [CFE_TBL_LoadCmd_Payload](#) [CFE_TBL_LoadCmd_Payload_t](#)
Load Table Command Payload.
- typedef struct [CFE_TBL_DumpCmd_Payload](#) [CFE_TBL_DumpCmd_Payload_t](#)
Dump Table Command Payload.
- typedef struct [CFE_TBL_ValidateCmd_Payload](#) [CFE_TBL_ValidateCmd_Payload_t](#)
Validate Table Command Payload.
- typedef struct [CFE_TBL_ActivateCmd_Payload](#) [CFE_TBL_ActivateCmd_Payload_t](#)
Activate Table Command Payload.
- typedef struct [CFE_TBL_DumpRegistryCmd_Payload](#) [CFE_TBL_DumpRegistryCmd_Payload_t](#)
Dump Registry Command Payload.
- typedef struct [CFE_TBL_SendRegistryCmd_Payload](#) [CFE_TBL_SendRegistryCmd_Payload_t](#)
Send Table Registry Command Payload.
- typedef struct [CFE_TBL_DelCDSCmd_Payload](#) [CFE_TBL_DelCDSCmd_Payload_t](#)
Delete Critical Table CDS Command Payload.
- typedef struct [CFE_TBL_AbortLoadCmd_Payload](#) [CFE_TBL_AbortLoadCmd_Payload_t](#)

Abort Load Command Payload.

- **typedef struct CFE_TBL_NotifyCmd_Payload** [CFE_TBL_NotifyCmd_Payload_t](#)
Table Management Notification Command Payload.
- **typedef struct CFE_TBL_HousekeepingTlm_Payload** [CFE_TBL_HousekeepingTlm_Payload_t](#)
- **typedef struct CFE_TBL_TblRegPacket_Payload** [CFE_TBL_TblRegPacket_Payload_t](#)

12.150.1 Detailed Description

Specification for the CFE Event Services (CFE_TBL) command and telemetry message constant definitions.
For CFE_TBL this is only the function/command code definitions

12.150.2 Typedef Documentation

12.150.2.1 CFE_TBL_AbortLoadCmd_Payload_t [typedef struct CFE_TBL_AbortLoadCmd_Payload](#) [CFE_TBL_AbortLoadCmd_Pay](#)
Abort Load Command Payload.

For command details, see [CFE_TBL_ABORT_LOAD_CC](#)

12.150.2.2 CFE_TBL_ActivateCmd_Payload_t [typedef struct CFE_TBL_ActivateCmd_Payload](#) [CFE_TBL_ActivateCmd_Pay](#)
Activate Table Command Payload.

For command details, see [CFE_TBL_ACTIVATE_CC](#)

12.150.2.3 CFE_TBL_DelCDSCmd_Payload_t [typedef struct CFE_TBL_DelCDSCmd_Payload](#) [CFE_TBL_DelCDSCmd_Pay](#)
Delete Critical Table CDS Command Payload.

For command details, see [CFE_TBL_DELETE_CDS_CC](#)

12.150.2.4 CFE_TBL_DumpCmd_Payload_t [typedef struct CFE_TBL_DumpCmd_Payload](#) [CFE_TBL_DumpCmd_Pay](#)
Dump Table Command Payload.

For command details, see [CFE_TBL_DUMP_CC](#)

12.150.2.5 CFE_TBL_DumpRegistryCmd_Payload_t [typedef struct CFE_TBL_DumpRegistryCmd_Payload](#) [CFE_TBL_DumpRegistryCmd_Pay](#)
[CFE_TBL_DumpRegistryCmd_Payload_t](#)

Dump Registry Command Payload.

For command details, see [CFE_TBL_DUMP_REGISTRY_CC](#)

12.150.2.6 CFE_TBL_HousekeepingTlm_Payload_t [typedef struct CFE_TBL_HousekeepingTlm_Payload](#) [CFE_TBL_HousekeepingTlm_Pay](#)
[CFE_TBL_HousekeepingTlm_Payload_t](#)

Name Table Services Housekeeping Packet

12.150.2.7 CFE_TBL_LoadCmd_Payload_t [typedef struct CFE_TBL_LoadCmd_Payload](#) [CFE_TBL_LoadCmd_Pay](#)
Load Table Command Payload.

For command details, see [CFE_TBL_LOAD_CC](#)

12.150.2.8 CFE_TBL_NotifyCmd_Payload_t [typedef struct CFE_TBL_NotifyCmd_Payload](#) [CFE_TBL_NotifyCmd_Pay](#)
Table Management Notification Command Payload.

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).

12.150.2.9 CFE_TBL_SendRegistryCmd_Payload_t [typedef struct CFE_TBL_SendRegistryCmd_Payload](#)

[CFE_TBL_SendRegistryCmd_Payload_t](#)

Send Table Registry Command Payload.

For command details, see [CFE_TBL_SEND_REGISTRY_CC](#)

12.150.2.10 CFE_TBL_TblRegPacket_Payload_t [typedef struct CFE_TBL_TblRegPacket_Payload](#) [CFE_TBL_TblRegPacket_Pay](#)

Name Table Registry Info Packet

12.150.2.11 CFE_TBL_ValidateCmd_Payload_t [typedef struct CFE_TBL_ValidateCmd_Payload](#) [CFE_TBL_ValidateCmd_Payload](#)

Validate Table Command Payload.

For command details, see [CFE_TBL_VALIDATE_CC](#)

12.151 cfe/modules/tbl/config/default_cfe_tbl_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_tbl_topicids.h"
```

Macros

- #define CFE_TBL_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_CMD_TOPICID)
/* 0x1804 */
- #define CFE_TBL_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_SEND_HK_TOPICID)
/* 0x180C */
- #define CFE_TBL_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_HK_TLM_TOPICID)
/* 0x0804 */
- #define CFE_TBL_REG_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_REG_TLM_TOPICID)
/* 0x080C */

12.151.1 Detailed Description

CFE Event Services (CFE_TBL) Application Message IDs

12.151.2 Macro Definition Documentation

12.151.2.1 CFE_TBL_CMD_MID #define CFE_TBL_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_CMD_TOPICID)

/* 0x1804 */

Definition at line 32 of file default_cfe_tbl_msgids.h.

12.151.2.2 CFE_TBL_HK_TLM_MID #define CFE_TBL_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_HK_TLM)

```
/* 0x0804 */
```

Definition at line 38 of file default_cfe_tbl_msgids.h.

12.151.2.3 CFE_TBL_REG_TLM_MID #define CFE_TBL_REG_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_REG)

```
/* 0x080C */
```

Definition at line 39 of file default_cfe_tbl_msgids.h.

12.151.2.4 CFE_TBL_SEND_HK_MID #define CFE_TBL_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_SEND)

```
/* 0x180C */
```

Definition at line 33 of file default_cfe_tbl_msgids.h.

12.152 cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h File Reference

```
#include "cfe_tbl_msgdefs.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- struct [CFE_TBL_NoopCmd](#)
- struct [CFE_TBL_ResetCountersCmd](#)
- struct [CFE_TBL_SendHkCmd](#)
- struct [CFE_TBL_LoadCmd](#)
Load Table Command.
- struct [CFE_TBL_DumpCmd](#)
- struct [CFE_TBL_ValidateCmd](#)
Validate Table Command.
- struct [CFE_TBL_ActivateCmd](#)
Activate Table Command.
- struct [CFE_TBL_DumpRegistryCmd](#)
Dump Registry Command.
- struct [CFE_TBL_SendRegistryCmd](#)
Send Table Registry Command.
- struct [CFE_TBL_DeleteCDSCmd](#)
Delete Critical Table CDS Command.
- struct [CFE_TBL_AbortLoadCmd](#)
Abort Load Command.
- struct [CFE_TBL_NotifyCmd](#)
- struct [CFE_TBL_HousekeepingTlm](#)
- struct [CFE_TBL_TableRegistryTlm](#)

Typedefs

- typedef struct [CFE_TBL_NoopCmd](#) CFE_TBL_NoopCmd_t
- typedef struct [CFE_TBL_ResetCountersCmd](#) CFE_TBL_ResetCountersCmd_t
- typedef struct [CFE_TBL_SendHkCmd](#) CFE_TBL_SendHkCmd_t
- typedef struct [CFE_TBL_LoadCmd](#) CFE_TBL_LoadCmd_t
Load Table Command.

- typedef struct [CFE_TBL_DumpCmd](#) [CFE_TBL_DumpCmd_t](#)
- typedef struct [CFE_TBL_ValidateCmd](#) [CFE_TBL_ValidateCmd_t](#)
Validate Table Command.
- typedef struct [CFE_TBL_ActivateCmd](#) [CFE_TBL_ActivateCmd_t](#)
Activate Table Command.
- typedef struct [CFE_TBL_DumpRegistryCmd](#) [CFE_TBL_DumpRegistryCmd_t](#)
Dump Registry Command.
- typedef struct [CFE_TBL_SendRegistryCmd](#) [CFE_TBL_SendRegistryCmd_t](#)
Send Table Registry Command.
- typedef struct [CFE_TBL_DeleteCDSCmd](#) [CFE_TBL_DeleteCDSCmd_t](#)
Delete Critical Table CDS Command.
- typedef struct [CFE_TBL_AbortLoadCmd](#) [CFE_TBL_AbortLoadCmd_t](#)
Abort Load Command.
- typedef struct [CFE_TBL_NotifyCmd](#) [CFE_TBL_NotifyCmd_t](#)
- typedef struct [CFE_TBL_HousekeepingTlm](#) [CFE_TBL_HousekeepingTlm_t](#)
- typedef struct [CFE_TBL_TableRegistryTlm](#) [CFE_TBL_TableRegistryTlm_t](#)

12.152.1 Detailed Description

Purpose: cFE Executive Services (TBL) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide
Notes:

12.152.2 Typedef Documentation

12.152.2.1 CFE_TBL_AbortLoadCmd_t [typedef struct CFE_TBL_AbortLoadCmd](#) [CFE_TBL_AbortLoadCmd_t](#)
Abort Load Command.

12.152.2.2 CFE_TBL_ActivateCmd_t [typedef struct CFE_TBL_ActivateCmd](#) [CFE_TBL_ActivateCmd_t](#)
Activate Table Command.

12.152.2.3 CFE_TBL_DeleteCDSCmd_t [typedef struct CFE_TBL_DeleteCDSCmd](#) [CFE_TBL_DeleteCDSCmd_t](#)
Delete Critical Table CDS Command.

12.152.2.4 CFE_TBL_DumpCmd_t [typedef struct CFE_TBL_DumpCmd](#) [CFE_TBL_DumpCmd_t](#)
/brief Dump Table Command

12.152.2.5 CFE_TBL_DumpRegistryCmd_t [typedef struct CFE_TBL_DumpRegistryCmd](#) [CFE_TBL_DumpRegistryCmd_t](#)
Dump Registry Command.

12.152.2.6 CFE_TBL_HousekeepingTlm_t [typedef struct CFE_TBL_HousekeepingTlm](#) [CFE_TBL_HousekeepingTlm_t](#)

12.152.2.7 CFE_TBL_LoadCmd_t `typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t`
Load Table Command.

12.152.2.8 CFE_TBL_NoopCmd_t `typedef struct CFE_TBL_NoopCmd CFE_TBL_NoopCmd_t`

12.152.2.9 CFE_TBL_NotifyCmd_t `typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t`
/brief Table Management Notification Command

12.152.2.10 CFE_TBL_ResetCountersCmd_t `typedef struct CFE_TBL_ResetCountersCmd CFE_TBL_ResetCountersCmd_t`

12.152.2.11 CFE_TBL_SendHkCmd_t `typedef struct CFE_TBL_SendHkCmd CFE_TBL_SendHkCmd_t`

12.152.2.12 CFE_TBL_SendRegistryCmd_t `typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t`
Send Table Registry Command.

12.152.2.13 CFE_TBL_TableRegistryTlm_t `typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t`

12.152.2.14 CFE_TBL_ValidateCmd_t `typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t`
Validate Table Command.

12.153 cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h File Reference

```
#include "cfe_tbl_mission_cfg.h"
#include "cfe_tbl_internal_cfg.h"
```

12.153.1 Detailed Description

CFE Table Services (CFE_TBL) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.154 cfe/modules/tbl/config/default_cfe_tbl_topicids.h File Reference

Macros

- `#define CFE_MISSION_TBL_CMD_TOPICID 4`
- `#define CFE_MISSION_TBL_SEND_HK_TOPICID 12`
- `#define CFE_MISSION_TBL_HK_TLM_TOPICID 4`
- `#define CFE_MISSION_TBL_REG_TLM_TOPICID 12`

12.154.1 Detailed Description

CFE Table Services (CFE_TBL) Application Topic IDs

12.154.2 Macro Definition Documentation

12.154.2.1 CFE_MISSION_TBL_CMD_TOPICID #define CFE_MISSION_TBL_CMD_TOPICID 4

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE command messages

Limits

Not Applicable

Definition at line 35 of file default_cfe_tbl_topicids.h.

12.154.2.2 CFE_MISSION_TBL_HK_TLM_TOPICID #define CFE_MISSION_TBL_HK_TLM_TOPICID 4

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE telemetry messages

Limits

Not Applicable

Definition at line 47 of file default_cfe_tbl_topicids.h.

12.154.2.3 CFE_MISSION_TBL_REG_TLM_TOPICID #define CFE_MISSION_TBL_REG_TLM_TOPICID 12

Definition at line 48 of file default_cfe_tbl_topicids.h.

12.154.2.4 CFE_MISSION_TBL_SEND_HK_TOPICID #define CFE_MISSION_TBL_SEND_HK_TOPICID 12

Definition at line 48 of file default_cfe_tbl_topicids.h.

12.155 cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h File Reference

Macros

TBL event IDs

- #define CFE_TBL_INIT_INF_EID 1
TB Initialization Event ID.
- #define CFE_TBL_NOOP_INF_EID 10
TBL No-op Command Success Event ID.
- #define CFE_TBL_RESET_INF_EID 11

- #define `CFE_TBL_FILE_LOADED_INF_EID` 12
TBL Load Table Command Success Event ID.
- #define `CFE_TBL_OVERWRITE_DUMP_INF_EID` 13
TBL Write Table To Existing File Success Event ID.
- #define `CFE_TBL_WRITE_DUMP_INF_EID` 14
TBL Write Table To New File Success Event ID.
- #define `CFE_TBL_OVERWRITE_REG_DUMP_INF_EID` 15
TBL Write Table Registry To Existing File Success Event ID.
- #define `CFE_TBL_VAL_REQ_MADE_INF_EID` 16
TBL Validate Table Request Success Event ID.
- #define `CFE_TBL_LOAD_PEND_REQ_INF_EID` 17
TBL Load Table Pending Notification Success Event ID.
- #define `CFE_TBL_TLM_REG_CMD_INF_EID` 18
TBL Telemeter Table Registry Entry Command Success Event ID.
- #define `CFE_TBL_LOAD_ABORT_INF_EID` 21
TBL Abort Table Load Success Event ID.
- #define `CFE_TBL_WRITE_REG_DUMP_INF_EID` 22
TBL Write Table Registry To New File Success Event ID.
- #define `CFE_TBL_ASSUMED_VALID_INF_EID` 23
TBL Validate Table Valid Due To No Validation Function Event ID.
- #define `CFE_TBL_LOAD_SUCCESS_INF_EID` 35
TBL Load Table API Success Event ID.
- #define `CFE_TBL_VALIDATION_INF_EID` 36
TBL Validate Table Success Event ID.
- #define `CFE_TBL_UPDATE_SUCCESS_INF_EID` 37
TBL Update Table Success Event ID.
- #define `CFE_TBL_CDS_DELETED_INFO_EID` 38
TBL Delete Table CDS Command Success Event ID.
- #define `CFE_TBL_MID_ERR_EID` 50
TBL Invalid Message ID Received Event ID.
- #define `CFE_TBL_CC1_ERR_EID` 51
TBL Invalid Command Code Received Event ID.
- #define `CFE_TBL_LEN_ERR_EID` 52
TBL Invalid Command Length Event ID.
- #define `CFE_TBL_FILE_ACCESS_ERR_EID` 53
TBL Load Table File Open Failure Event ID.
- #define `CFE_TBL_FILE_STD_HDR_ERR_EID` 54
TBL Load Table File Read Standard Header Failure Event ID.
- #define `CFE_TBL_FILE_TBL_HDR_ERR_EID` 55
TBL Load Table File Read Table Header Failure Event ID.
- #define `CFE_TBL_FAIL_HK_SEND_ERR_EID` 56
TBL Send Housekeeping Command Transmit Failure Event ID.
- #define `CFE_TBL_NO SUCH_TABLE_ERR_EID` 57
TBL Table Name Not Found Event ID.
- #define `CFE_TBL_FILE_TYPE_ERR_EID` 58
TBL Load Table Invalid File Content ID Event ID.
- #define `CFE_TBL_FILE_SUBTYPE_ERR_EID` 59
TBL Load Table Invalid File Subtype Event ID.
- #define `CFE_TBL_NO_WORK_BUFFERS_ERR_EID` 60
TBL Load Or Dump Table No Working Buffers Available Event ID.
- #define `CFE_TBL_CREATING_DUMP_FILE_ERR_EID` 62
TBL Write File Creation Failure Event ID.
- #define `CFE_TBL_WRITE_CFE_HDR_ERR_EID` 63
TBL Write Standard File Header Failure Event ID.

- #define `CFE_TBL_WRITE_TBL_HDR_ERR_EID` 64
TBL Write Table File Header Failure Event ID.
- #define `CFE_TBL_WRITE_TBL_IMG_ERR_EID` 65
TBL Write Table File Data Failure Event ID.
- #define `CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID` 66
TBL Validate Or Write Table Command No Inactive Buffer Event ID.
- #define `CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID` 67
TBL Validate Table Command Result Storage Exceeded Event ID.
- #define `CFE_TBL_WRITE_TBL_REG_ERR_EID` 68
TBL Write Table Registry File Data Failure Event ID.
- #define `CFE_TBL_LOAD_ABORT_ERR_EID` 69
TBL Abort Table Load No Load Started Event ID.
- #define `CFE_TBL_ACTIVATE_ERR_EID` 70
TBL Activate Table Command No Inactive Buffer Event ID.
- #define `CFE_TBL_FILE_INCOMPLETE_ERR_EID` 71
TBL Load Table Incomplete Load Event ID.
- #define `CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID` 72
TBL Load Table File Exceeds Table Size Event ID.
- #define `CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID` 73
TBL Load Table File Zero Length Event ID.
- #define `CFE_TBL_PARTIAL_LOAD_ERR_EID` 74
TBL Load Table Uninitialized Partial Load Event ID.
- #define `CFE_TBL_FILE_TOO_BIG_ERR_EID` 75
TBL Load Table File Excess Data Event ID.
- #define `CFE_TBL_TOO_MANY_DUMPS_ERR_EID` 76
TBL Write Table Command Dump Only Control Blocks Exceeded Event ID.
- #define `CFE_TBL_DUMP_PENDING_ERR_EID` 77
TBL Write Table Command Already In Progress Event ID.
- #define `CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID` 78
TBL Activate Table Command For Dump Only Table Event ID.
- #define `CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID` 79
TBL Load Table For Dump Only Table Event ID.
- #define `CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID` 80
TBL Validate Or Write Table Command Invalid Buffer Event ID.
- #define `CFE_TBL_UNVALIDATED_ERR_EID` 81
TBL Activate Table Command Inactive Image Not Validated Event ID.
- #define `CFE_TBL_IN_REGISTRY_ERR_EID` 82
TBL Delete Table CDS Command For Registered Table Event ID.
- #define `CFE_TBL_NOT_CRITICAL_TBL_ERR_EID` 83
TBL Delete Table CDS Command Invalid CDS Type Event ID.
- #define `CFE_TBL_NOT_IN_CRIT_REG_ERR_EID` 84
TBL Delete Table CDS Command Not In Critical Table Registry Event ID.
- #define `CFE_TBL_CDS_NOT_FOUND_ERR_EID` 85
TBL Delete Table CDS Command Not In CDS Registry Event ID.
- #define `CFE_TBL_CDS_DELETE_ERR_EID` 86
TBL Delete Table CDS Command Internal Error Event ID.
- #define `CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID` 87
TBL Delete Table CDS Command App Active Event ID.
- #define `CFE_TBL_LOADING_PENDING_ERR_EID` 88
TBL Load Table Command Load Pending Event ID.
- #define `CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID` 89
TBL Send Notification Transmit Failed Event ID.
- #define `CFE_TBL_REGISTER_ERR_EID` 90
TBL Register Table Failed Event ID.
- #define `CFE_TBL_SHARE_ERR_EID` 91

- #define [CFE_TBL_UNREGISTER_ERR_EID](#) 92
TBL Unregister Table Failed Event ID.
- #define [CFE_TBL_LOAD_VAL_ERR_EID](#) 93
TBL Validation Function Invalid Return Code Event ID.
- #define [CFE_TBL_LOAD_TYPE_ERR_EID](#) 94
TBL Load Table API Invalid Source Type Event ID.
- #define [CFE_TBL_UPDATE_ERR_EID](#) 95
TBL Update Table Failed Event ID.
- #define [CFE_TBL_VALIDATION_ERR_EID](#) 96
TBL Validate Table Validation Failed Event ID.
- #define [CFE_TBL_SPACECRAFT_ID_ERR_EID](#) 97
TBL Read Header Invalid Spacecraft ID Event ID.
- #define [CFE_TBL_PROCESSOR_ID_ERR_EID](#) 98
TBL Read Header Invalid Processor ID Event ID.
- #define [CFE_TBL_LOAD_IN_PROGRESS_ERR_EID](#) 100
TBL Load Table API Load Already In Progress Event ID.
- #define [CFE_TBL_LOAD_FILENAME_LONG_ERR_EID](#) 101
TBL Load Table Filename Too Long Event ID.
- #define [CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID](#) 102
TBL Load Table Name Mismatch Event ID.
- #define [CFE_TBL_HANDLE_ACCESS_ERR_EID](#) 103
TBL Load Table API Access Violation Event ID.

12.155.1 Detailed Description

cFE Table Services Event IDs

12.155.2 Macro Definition Documentation

12.155.2.1 CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID #define CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID 78
TBL Activate Table Command For Dump Only Table Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to table being dump only.
Definition at line 544 of file cfe_tbl_eventids.h.

12.155.2.2 CFE_TBL_ACTIVATE_ERR_EID #define CFE_TBL_ACTIVATE_ERR_EID 70
TBL Activate Table Command No Inactive Buffer Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to no associated inactive buffer.
Definition at line 450 of file cfe_tbl_eventids.h.

12.155.2.3 CFE_TBL_ASSUMED_VALID_INF_EID #define CFE_TBL_ASSUMED_VALID_INF_EID 23
TBL Validate Table Valid Due To No Validation Function Event ID.

Type: INFORMATION

Cause:

[TBL Validate Table Command](#) marking table as valid due to no validation function being registered.
Definition at line 180 of file cfe_tbl_eventids.h.

12.155.2.4 CFE_TBL_CC1_ERR_EID #define CFE_TBL_CC1_ERR_EID 51
TBL Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE_TBL_CMD_MID](#) received on the TBL message pipe.
Definition at line 246 of file cfe_tbl_eventids.h.

12.155.2.5 CFE_TBL_CDS_DELETE_ERR_EID #define CFE_TBL_CDS_DELETE_ERR_EID 86
TBL Delete Table CDS Command Internal Error Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to an internal error. See the system log for more information.
Definition at line 640 of file cfe_tbl_eventids.h.

12.155.2.6 CFE_TBL_CDS_DELETED_INFO_EID #define CFE_TBL_CDS_DELETED_INFO_EID 38
TBL Delete Table CDS Command Success Event ID.

Type: INFORMATION

Cause:

[TBL Delete Table CDS Command](#) success.
Definition at line 224 of file cfe_tbl_eventids.h.

12.155.2.7 CFE_TBL_CDS_NOT_FOUND_ERR_EID #define CFE_TBL_CDS_NOT_FOUND_ERR_EID 85
TBL Delete Table CDS Command Not In CDS Registry Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table name not found in the CDS registry.
Definition at line 628 of file cfe_tbl_eventids.h.

12.155.2.8 CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID #define CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID 87
TBL Delete Table CDS Command App Active Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the owning application being active.
Definition at line 652 of file cfe_tbl_eventids.h.

12.155.2.9 CFE_TBL_CREATING_DUMP_FILE_ERR_EID #define CFE_TBL_CREATING_DUMP_FILE_ERR_EID 62
TBL Write File Creation Failure Event ID.

Type: ERROR

Cause:

TBL Write Table or Table Registry File failed to create file. OVERLOADED
Definition at line 357 of file cfe_tbl_eventids.h.

12.155.2.10 CFE_TBL_DUMP_PENDING_ERR_EID #define CFE_TBL_DUMP_PENDING_ERR_EID 77
TBL Write Table Command Already In Progress Event ID.

Type: ERROR

Cause:

[TBL Write Table Command](#) failure due to a dump already in progress for the same table.
Definition at line 532 of file cfe_tbl_eventids.h.

12.155.2.11 CFE_TBL_FAIL_HK_SEND_ERR_EID #define CFE_TBL_FAIL_HK_SEND_ERR_EID 56
TBL Send Housekeeping Command Transmit Failure Event ID.

Type: ERROR

Cause:

[TBL Send Housekeeping Command](#) failure transmitting the housekeeping message.
Definition at line 302 of file cfe_tbl_eventids.h.

12.155.2.12 CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID #define CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID 89
TBL Send Notification Transmit Failed Event ID.

Type: ERROR

Cause:

TBL send notification transmit message failure.
Definition at line 674 of file cfe_tbl_eventids.h.

12.155.2.13 CFE_TBL_FILE_ACCESS_ERR_EID #define CFE_TBL_FILE_ACCESS_ERR_EID 53
TBL Load Table File Open Failure Event ID.

Type: ERROR

Cause:

Load Table failure opening the file. OVERLOADED
Definition at line 268 of file cfe_tbl_eventids.h.

12.155.2.14 CFE_TBL_FILE_INCOMPLETE_ERR_EID #define CFE_TBL_FILE_INCOMPLETE_ERR_EID 71
TBL Load Table Incomplete Load Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to inability to read the size of data specified in the table header from file. OVERLOADED
Definition at line 462 of file cfe_tbl_eventids.h.

12.155.2.15 CFE_TBL_FILE_LOADED_INF_EID #define CFE_TBL_FILE_LOADED_INF_EID 12
TBL Load Table Command Success Event ID.

Type: INFORMATION

Cause:

[TBL Load Table Command](#) successfully loaded the new table data to the working buffer.
Definition at line 76 of file cfe_tbl_eventids.h.

12.155.2.16 CFE_TBL_FILE_STD_HDR_ERR_EID #define CFE_TBL_FILE_STD_HDR_ERR_EID 54
TBL Load Table File Read Standard Header Failure Event ID.

Type: ERROR

Cause:

Load Table failure reading the file standard header.
Definition at line 279 of file cfe_tbl_eventids.h.

12.155.2.17 CFE_TBL_FILE_SUBTYPE_ERR_EID #define CFE_TBL_FILE_SUBTYPE_ERR_EID 59
TBL Load Table Invalid File Subtype Event ID.

Type: ERROR

Cause:

TBL Load Table Failure due to invalid file subtype.
Definition at line 335 of file cfe_tbl_eventids.h.

12.155.2.18 CFE_TBL_FILE_TBL_HDR_ERR_EID #define CFE_TBL_FILE_TBL_HDR_ERR_EID 55
TBL Load Table File Read Table Header Failure Event ID.

Type: ERROR

Cause:

Load Table failure reading the file table header.
Definition at line 290 of file cfe_tbl_eventids.h.

12.155.2.19 CFE_TBL_FILE_TOO_BIG_ERR_EID #define CFE_TBL_FILE_TOO_BIG_ERR_EID 75
TBL Load Table File Excess Data Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified size of data being smaller than the actual data contained in the file. OVERLOADED

Definition at line 508 of file cfe_tbl_eventids.h.

12.155.2.20 CFE_TBL_FILE_TYPE_ERR_EID #define CFE_TBL_FILE_TYPE_ERR_EID 58
TBL Load Table Invalid File Content ID Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to invalid file content ID.

Definition at line 324 of file cfe_tbl_eventids.h.

12.155.2.21 CFE_TBL_HANDLE_ACCESS_ERR_EID #define CFE_TBL_HANDLE_ACCESS_ERR_EID 103
TBL Load Table API Access Violation Event ID.

Type: ERROR

Cause:

[CFE_TBL_Load](#) API failure due to the application not owning the table.

Definition at line 817 of file cfe_tbl_eventids.h.

12.155.2.22 CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID #define CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID 80
TBL Validate Or Write Table Command Invalid Buffer Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) or [TBL Write Table Command](#) failure due to an invalid buffer selection. OVERLOADED
Definition at line 568 of file cfe_tbl_eventids.h.

12.155.2.23 CFE_TBL_IN_REGISTRY_ERR_EID #define CFE_TBL_IN_REGISTRY_ERR_EID 82
TBL Delete Table CDS Command For Registered Table Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table being currently registered.
Definition at line 592 of file cfe_tbl_eventids.h.

12.155.2.24 CFE_TBL_INIT_INF_EID #define CFE_TBL_INIT_INF_EID 1
TB Initialization Event ID.

Type: INFORMATION

Cause:

Table Services Task initialization complete.
Definition at line 42 of file cfe_tbl_eventids.h.

12.155.2.25 CFE_TBL_LEN_ERR_EID #define CFE_TBL_LEN_ERR_EID 52
TBL Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the message ID and command code received on the TBL message pipe.
Definition at line 257 of file cfe_tbl_eventids.h.

12.155.2.26 CFE_TBL_LOAD_ABORT_ERR_EID #define CFE_TBL_LOAD_ABORT_ERR_EID 69
TBL Abort Table Load No Load Started Event ID.

Type: ERROR

Cause:

[TBL Abort Table Load Command](#) failure due to no load in progress.
Definition at line 438 of file cfe_tbl_eventids.h.

12.155.2.27 CFE_TBL_LOAD_ABORT_INF_EID #define CFE_TBL_LOAD_ABORT_INF_EID 21
TBL Abort Table Load Success Event ID.

Type: INFORMATION

Cause:

[TBL Abort Table Load Command](#) success.

Definition at line 157 of file cfe_tbl_eventids.h.

12.155.2.28 CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID #define CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID 72
TBL Load Table File Exceeds Table Size Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified offset and/or size of data exceeding the table size. OVERLOADED
Definition at line 474 of file cfe_tbl_eventids.h.

12.155.2.29 CFE_TBL_LOAD_FILENAME_LONG_ERR_EID #define CFE_TBL_LOAD_FILENAME_LONG_ERR_EID 101
TBL Load Table Filename Too Long Event ID.

Type: ERROR

Cause:

Load table filename too long.

Definition at line 795 of file cfe_tbl_eventids.h.

12.155.2.30 CFE_TBL_LOAD_IN_PROGRESS_ERR_EID #define CFE_TBL_LOAD_IN_PROGRESS_ERR_EID 100
TBL Load Table API Load Already In Progress Event ID.

Type: ERROR

Cause:

[CFE_TBL_Load](#) API failure due to load already in progress.

Definition at line 784 of file cfe_tbl_eventids.h.

12.155.2.31 CFE_TBL_LOAD_PEND_REQ_INF_EID #define CFE_TBL_LOAD_PEND_REQ_INF_EID 17
TBL Load Table Pending Notification Success Event ID.

Type: DEBUG

Cause:

TBL load table pending notification successfully sent.
Definition at line 134 of file cfe_tbl_eventids.h.

12.155.2.32 CFE_TBL_LOAD_SUCCESS_INF_EID #define CFE_TBL_LOAD_SUCCESS_INF_EID 35
TBL Load Table API Success Event ID.

Type: DEBUG (the first time) and INFORMATION (normally)

Cause:

[CFE_TBL_Load](#) API success for dump only or normal table. OVERLOADED
Definition at line 191 of file cfe_tbl_eventids.h.

12.155.2.33 CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID #define CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID 102
TBL Load Table Name Mismatch Event ID.

Type: ERROR

Cause:

Load table name in the table file header does not match the specified table name.
Definition at line 806 of file cfe_tbl_eventids.h.

12.155.2.34 CFE_TBL_LOAD_TYPE_ERR_EID #define CFE_TBL_LOAD_TYPE_ERR_EID 94
TBL Load Table API Invalid Source Type Event ID.

Type: ERROR

Cause:

[CFE_TBL_Load](#) API valid due to invalid source type.
Definition at line 729 of file cfe_tbl_eventids.h.

12.155.2.35 CFE_TBL_LOAD_VAL_ERR_EID #define CFE_TBL_LOAD_VAL_ERR_EID 93
TBL Validation Function Invalid Return Code Event ID.

Type: ERROR

Cause:

Invalid table validation function return code.
Definition at line 718 of file cfe_tbl_eventids.h.

12.155.2.36 CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID #define CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID 79
TBL Load Table For Dump Only Table Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to table being dump only. OVERLOADED
Definition at line 555 of file cfe_tbl_eventids.h.

12.155.2.37 CFE_TBL_LOADING_PENDING_ERR_EID #define CFE_TBL_LOADING_PENDING_ERR_EID 88
TBL Load Table Command Load Pending Event ID.

Type: ERROR

Cause:

[TBL Load Table Command](#) failed due to a load already pending.
Definition at line 663 of file cfe_tbl_eventids.h.

12.155.2.38 CFE_TBL_MID_ERR_EID #define CFE_TBL_MID_ERR_EID 50
TBL Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the TBL message pipe.
Definition at line 235 of file cfe_tbl_eventids.h.

12.155.2.39 CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID #define CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID 66
TBL Validate Or Write Table Command No Inactive Buffer Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) or [TBL Write Table Command](#) failure due to requesting non-existent inactive buffer. OVERLOADED
Definition at line 403 of file cfe_tbl_eventids.h.

12.155.2.40 CFE_TBL_NO SUCH_TABLE_ERR_EID #define CFE_TBL_NO SUCH_TABLE_ERR_EID 57
TBL Table Name Not Found Event ID.

Type: ERROR

Cause:

TBL command handler unable to find table name. OVERLOADED
Definition at line 313 of file cfe_tbl_eventids.h.

12.155.2.41 CFE_TBL_NO_WORK_BUFFERS_ERR_EID #define CFE_TBL_NO_WORK_BUFFERS_ERR_EID 60
TBL Load Or Dump Table No Working Buffers Available Event ID.

Type: ERROR

Cause:

TBL Load or Dump failure due to no working buffers available or internal error. OVERLOADED
Definition at line 346 of file cfe_tbl_eventids.h.

12.155.2.42 CFE_TBL_NOOP_INF_EID #define CFE_TBL_NOOP_INF_EID 10
TBL No-op Command Success Event ID.

Type: INFORMATION

Cause:

[NO-OP TBL No-op Command](#) success.
Definition at line 53 of file cfe_tbl_eventids.h.

12.155.2.43 CFE_TBL_NOT_CRITICAL_TBL_ERR_EID #define CFE_TBL_NOT_CRITICAL_TBL_ERR_EID 83
TBL Delete Table CDS Command Invalid CDS Type Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to CDS being in the table registry but not registered as a table within ES.
Definition at line 604 of file cfe_tbl_eventids.h.

12.155.2.44 CFE_TBL_NOT_IN_CRIT_REG_ERR_EID #define CFE_TBL_NOT_IN_CRIT_REG_ERR_EID 84
TBL Delete Table CDS Command Not In Critical Table Registry Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table not being in the critical table registry.
Definition at line 616 of file cfe_tbl_eventids.h.

12.155.2.45 CFE_TBL_OVERWRITE_DUMP_INF_EID #define CFE_TBL_OVERWRITE_DUMP_INF_EID 13
TBL Write Table To Existing File Success Event ID.

Type: INFORMATION

Cause:

TBL write table to an existing file success.
Definition at line 87 of file cfe_tbl_eventids.h.

12.155.2.46 CFE_TBL_OVERWRITE_REG_DUMP_INF_EID #define CFE_TBL_OVERWRITE_REG_DUMP_INF_EID 15
TBL Write Table Registry To Existing File Success Event ID.

Type: DEBUG

Cause:

TBL Write Table Registry to an existing file completed successfully.
Definition at line 109 of file cfe_tbl_eventids.h.

12.155.2.47 CFE_TBL_PARTIAL_LOAD_ERR_EID #define CFE_TBL_PARTIAL_LOAD_ERR_EID 74
TBL Load Table Uninitialized Partial Load Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to attempting a partial load to an uninitialized table. OVERLOADED
Definition at line 496 of file cfe_tbl_eventids.h.

12.155.2.48 CFE_TBL_PROCESSOR_ID_ERR_EID #define CFE_TBL_PROCESSOR_ID_ERR_EID 98
TBL Read Header Invalid Processor ID Event ID.

Type: ERROR

Cause:

Invalid processor ID in table file header.
Definition at line 773 of file cfe_tbl_eventids.h.

12.155.2.49 CFE_TBL_REGISTER_ERR_EID #define CFE_TBL_REGISTER_ERR_EID 90
TBL Register Table Failed Event ID.

Type: ERROR

Cause:

TBL table registration failure. See system log for more information.
Definition at line 685 of file cfe_tbl_eventids.h.

12.155.2.50 CFE_TBL_RESET_INF_EID #define CFE_TBL_RESET_INF_EID 11
TBL Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[TBL Reset Counters Command](#) success.
Definition at line 64 of file cfe_tbl_eventids.h.

12.155.2.51 CFE_TBL_SHARE_ERR_EID #define CFE_TBL_SHARE_ERR_EID 91
TBL Share Table Failed Event ID.

Type: ERROR

Cause:

TBL share table failure. See system log for more information.
Definition at line 696 of file cfe_tbl_eventids.h.

12.155.2.52 CFE_TBL_SPACECRAFT_ID_ERR_EID #define CFE_TBL_SPACECRAFT_ID_ERR_EID 97
TBL Read Header Invalid Spacecraft ID Event ID.

Type: ERROR

Cause:

Invalid spacecraft ID in table file header.
Definition at line 762 of file cfe_tbl_eventids.h.

12.155.2.53 CFE_TBL_TLM_REG_CMD_INF_EID #define CFE_TBL_TLM_REG_CMD_INF_EID 18
TBL Telemeter Table Registry Entry Command Success Event ID.

Type: DEBUG

Cause:

[TBL Telemeter Table Registry Entry command](#) successfully set the table registry index to telemeter in the next house-keeping packet.
Definition at line 146 of file cfe_tbl_eventids.h.

12.155.2.54 CFE_TBL_TOO_MANY_DUMPS_ERR_EID #define CFE_TBL_TOO_MANY_DUMPS_ERR_EID 76
TBL Write Table Command Dump Only Control Blocks Exceeded Event ID.

Type: ERROR

Cause:

[TBL Write Table Command](#) failure due to exceeding the allocated number of control blocks available to write a dump only table.
Definition at line 520 of file cfe_tbl_eventids.h.

12.155.2.55 CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID #define CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID 67

TBL Validate Table Command Result Storage Exceeded Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) failure due to exceeding result storage.

Definition at line 415 of file cfe_tbl_eventids.h.

12.155.2.56 CFE_TBL_UNREGISTER_ERR_EID #define CFE_TBL_UNREGISTER_ERR_EID 92

TBL Unregister Table Failed Event ID.

Type: ERROR

Cause:

TBL unregister table failure. See system log for more information.

Definition at line 707 of file cfe_tbl_eventids.h.

12.155.2.57 CFE_TBL_UNVALIDATED_ERR_EID #define CFE_TBL_UNVALIDATED_ERR_EID 81

TBL Activate Table Command Inactive Image Not Validated Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to the inactive image not being validated.

Definition at line 580 of file cfe_tbl_eventids.h.

12.155.2.58 CFE_TBL_UPDATE_ERR_EID #define CFE_TBL_UPDATE_ERR_EID 95

TBL Update Table Failed Event ID.

Type: ERROR

Cause:

TBL update table failure due to an internal error. OVERLOADED

Definition at line 740 of file cfe_tbl_eventids.h.

12.155.2.59 CFE_TBL_UPDATE_SUCCESS_INF_EID #define CFE_TBL_UPDATE_SUCCESS_INF_EID 37
TBL Update Table Success Event ID.

Type: INFORMATION

Cause:

Table update successfully completed.

Definition at line 213 of file cfe_tbl_eventids.h.

12.155.2.60 CFE_TBL_VAL_REQ_MADE_INF_EID #define CFE_TBL_VAL_REQ_MADE_INF_EID 16
TBL Validate Table Request Success Event ID.

Type: DEBUG

Cause:

[TBL Validate Table Command](#) success. Note this event signifies the request to validate the table has been successfully submitted. Completion will generate a [CFE_TBL_VALIDATION_INF_EID](#) or [CFE_TBL_VALIDATION_ERR_EID](#) event messages.

Definition at line 123 of file cfe_tbl_eventids.h.

12.155.2.61 CFE_TBL_VALIDATION_ERR_EID #define CFE_TBL_VALIDATION_ERR_EID 96
TBL Validate Table Validation Failed Event ID.

Type: ERROR

Cause:

TBL validate table function indicates validation failed. OVERLOADED

Definition at line 751 of file cfe_tbl_eventids.h.

12.155.2.62 CFE_TBL_VALIDATION_INF_EID #define CFE_TBL_VALIDATION_INF_EID 36
TBL Validate Table Success Event ID.

Type: INFORMATION

Cause:

Table active or inactive image successfully validated by the registered validation function. OVERLOADED

Definition at line 202 of file cfe_tbl_eventids.h.

12.155.2.63 CFE_TBL_WRITE_CFE_HDR_ERR_EID #define CFE_TBL_WRITE_CFE_HDR_ERR_EID 63
TBL Write Standard File Header Failure Event ID.

Type: ERROR

Cause:

TBL Write Table or Table Registry File failure writing the standard file header. OVERLOADED
Definition at line 368 of file cfe_tbl_eventids.h.

12.155.2.64 CFE_TBL_WRITE_DUMP_INF_EID #define CFE_TBL_WRITE_DUMP_INF_EID 14
TBL Write Table To New File Success Event ID.

Type: INFORMATION

Cause:

TBL write table to a new file success.
Definition at line 98 of file cfe_tbl_eventids.h.

12.155.2.65 CFE_TBL_WRITE_REG_DUMP_INF_EID #define CFE_TBL_WRITE_REG_DUMP_INF_EID 22
TBL Write Table Registry To New File Success Event ID.

Type: DEBUG

Cause:

TBL Write Table Registry to a new file completed successfully.
Definition at line 168 of file cfe_tbl_eventids.h.

12.155.2.66 CFE_TBL_WRITE_TBL_HDR_ERR_EID #define CFE_TBL_WRITE_TBL_HDR_ERR_EID 64
TBL Write Table File Header Failure Event ID.

Type: ERROR

Cause:

TBL Write Table failure writing the table image file header.
Definition at line 379 of file cfe_tbl_eventids.h.

12.155.2.67 CFE_TBL_WRITE_TBL_IMG_ERR_EID #define CFE_TBL_WRITE_TBL_IMG_ERR_EID 65
TBL Write Table File Data Failure Event ID.

Type: ERROR

Cause:

TBL Write Table failure writing the table data.
Definition at line 390 of file cfe_tbl_eventids.h.

12.155.2.68 CFE_TBL_WRITE_TBL_REG_ERR_EID #define CFE_TBL_WRITE_TBL_REG_ERR_EID 68
TBL Write Table Registry File Data Failure Event ID.

Type: ERROR

Cause:

TB Write Table Registry failure writing file data.
Definition at line 426 of file cfe_tbl_eventids.h.

12.155.2.69 CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID #define CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID 73
TBL Load Table File Zero Length Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified size of data being zero.
Definition at line 485 of file cfe_tbl_eventids.h.

12.156 cfe/modules/time/config/default_cfe_time_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Data Structures

- struct [CFE_TIME_SysTime](#)

Data structure used to hold system time values.

Typedefs

- `typedef struct CFE_TIME_SysTime CFE_TIME_SysTime_t`
Data structure used to hold system time values.
- `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_TIME_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,
 CFE_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_EXTERNAL
 = 2 }`
Label definitions associated with CFE_TIME_SourceSelect_Enum_t.
- `enum CFE_TIME_ToneSignalSelect { CFE_TIME_ToneSignalSelect_PRIMARY = 1, CFE_TIME_ToneSignalSelect_REDUNDANT
 = 2 }`
Label definitions associated with CFE_TIME_ToneSignalSelect_Enum_t.
- `enum CFE_TIME_AdjustDirection { CFE_TIME_AdjustDirection_ADD = 1, CFE_TIME_AdjustDirection_SUBTRACT
 = 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.

12.156.1 Detailed Description

Declarations and prototypes for cfe_time_extern_typedefs module

12.156.2 Typedef Documentation

12.156.2.1 CFE_TIME_AdjustDirection_Enum_t `typedef uint8 CFE_TIME_AdjustDirection_Enum_t`
STCF adjustment direction (for both one-time and 1Hz adjustments)

See also

enum [CFE_TIME_AdjustDirection](#)

Definition at line 234 of file default_cfe_time_extern_typedefs.h.

12.156.2.2 CFE_TIME_ClockState_Enum_t `typedef int16 CFE_TIME_ClockState_Enum_t`
Enumerated types identifying the quality of the current time.

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 considered 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 165 of file default_cfe_time_extern_typedefs.h.

12.156.2.3 CFE_TIME_FlagBit_Enum_t `typedef uint8 CFE_TIME_FlagBit_Enum_t`
Bit positions of the various clock state flags.

See also

enum [CFE_TIME_FlagBit](#)

Definition at line 113 of file default_cfe_time_extern_typedefs.h.

12.156.2.4 CFE_TIME_FlywheelState_Enum_t `typedef uint8 CFE_TIME_FlywheelState_Enum_t`
Fly-wheel status values.

See also

enum [CFE_TIME_FlywheelState](#)

Definition at line 257 of file default_cfe_time_extern_typedefs.h.

12.156.2.5 CFE_TIME_SetState_Enum_t `typedef uint8 CFE_TIME_SetState_Enum_t`
Clock status values (has the clock been set to correct time)

See also

enum [CFE_TIME_SetState](#)

Definition at line 280 of file default_cfe_time_extern_typedefs.h.

12.156.2.6 CFE_TIME_SourceSelect_Enum_t `typedef uint8 CFE_TIME_SourceSelect_Enum_t`
Clock Source Selection Parameters.

See also

enum [CFE_TIME_SourceSelect](#)

Definition at line 188 of file default_cfe_time_extern_typedefs.h.

12.156.2.7 CFE_TIME_SysTime_t `typedef struct CFE_TIME_SysTime CFE_TIME_SysTime_t`
Data structure used to hold system time values.

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.

12.156.2.8 CFE_TIME_ToneSignalSelect_Enum_t `typedef uint8 CFE_TIME_ToneSignalSelect_Enum_t`
Tone Signal Selection Parameters.

See also

enum [CFE_TIME_ToneSignalSelect](#)

Definition at line 211 of file default_cfe_time_extern_typedefs.h.

12.156.3 Enumeration Type Documentation

12.156.3.1 CFE_TIME_AdjustDirection `enum CFE_TIME_AdjustDirection`
Label definitions associated with `CFE_TIME_AdjustDirection_Enum_t`.

Enumerator

<code>CFE_TIME_AdjustDirection_ADD</code>	Add time adjustment.
<code>CFE_TIME_AdjustDirection_SUBTRACT</code>	Subtract time adjustment.

Definition at line 216 of file default_cfe_time_extern_typedefs.h.

12.156.3.2 CFE_TIME_ClockState `enum CFE_TIME_ClockState`
Label definitions associated with `CFE_TIME_ClockState_Enum_t`.

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 118 of file default_cfe_time_extern_typedefs.h.

12.156.3.3 CFE_TIME_FlagBit enum CFE_TIME_FlagBit

Label definitions associated with CFE_TIME_FlagBit_Enum_t.

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 50 of file default_cfe_time_extern_typedefs.h.

12.156.3.4 CFE_TIME_FlywheelState enum CFE_TIME_FlywheelState

Label definitions associated with CFE_TIME_FlywheelState_Enum_t.

Enumerator

CFE_TIME_FlywheelState_NO_FLY	Not in flywheel state.
CFE_TIME_FlywheelState_IS_FLY	In flywheel state.

Definition at line 239 of file default_cfe_time_extern_typedefs.h.

12.156.3.5 CFE_TIME_SetState enum CFE_TIME_SetState

Label definitions associated with CFE_TIME_SetState_Enum_t.

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 262 of file default_cfe_time_extern_typedefs.h.

12.156.3.6 CFE_TIME_SourceSelect enum CFE_TIME_SourceSelect

Label definitions associated with CFE_TIME_SourceSelect_Enum_t.

Enumerator

CFE_TIME_SourceSelect_INTERNAL	Use Internal Source.
CFE_TIME_SourceSelect_EXTERNAL	Use External Source.

Definition at line 170 of file default_cfe_time_extern_typedefs.h.

12.156.3.7 CFE_TIME_ToneSignalSelect enum CFE_TIME_ToneSignalSelect

Label definitions associated with CFE_TIME_ToneSignalSelect_Enum_t.

Enumerator

CFE_TIME_ToneSignalSelect_PRIMARY	Primary Source.
CFE_TIME_ToneSignalSelect_REDUNDANT	Redundant Source.

Definition at line 193 of file default_cfe_time_extern_typedefs.h.

12.157 cfe/modules/time/config/default_cfe_time_fcncodes.h File Reference**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_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_ONE_HZ_ADJUSTMENT_CC 13 /* add 1Hz STCF adjustment */
- #define CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC 14 /* subtract 1Hz STCF adjustment */
- #define CFE_TIME_SET_SIGNAL_CC 15 /* set clock signal (pri vs red) */

12.157.1 Detailed Description

Specification for the CFE Time Services (CFE_TIME) command function codes

Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

12.157.2 Macro Definition Documentation

12.157.2.1 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_AddSTCFAadj

Command Structure

[CFE_TIME_AddAdjustCmd_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)
- Platform 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_ONE_HZ_ADJUSTMENT_CC](#),
[CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#)

Definition at line 521 of file default_cfe_time_fcncodes.h.

12.157.2.2 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_AddDelayCmd_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_DLlatentS](#), [\\$sc_\\$cpu_TIME_DLlatentSs](#) - Housekeeping Telemetry point indicating command specified values
- [\\$sc_\\$cpu_TIME_DLlatentDir](#) - 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)
- Platform 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 290 of file default_cfe_time_fcncodes.h.

```
12.157.2.3 CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC #define CFE_TIME_ADD_ONE_HZ_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_AddOneHzAdjustmentCmd_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_ONEHZ_EID](#) informational event message will be generated

Error Conditions

- Platform 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_ONEHZ_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_ONE_HZ_ADJUSTMENT_CC](#)

Definition at line 601 of file default_cfe_time_fcncodes.h.

12.157.2.4 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_NoopCmd_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 66 of file default_cfe_time_fcncodes.h.

12.157.2.5 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_ResetCountersCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_TIME_CMDPC** - command execution counter will reset to 0
- **\$sc_\$cpu_TIME_CMDEC** - command error counter will reset to 0
- 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 reset unconditionally.

Criticality

None

See also

Definition at line 111 of file default_cfe_time_fcncodes.h.

12.157.2.6 CFE_TIME_SEND_DIAGNOSTIC_CC #define CFE_TIME_SEND_DIAGNOSTIC_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_SendDiagnosticCmd_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 145 of file default_cfe_time_fcncodes.h.

12.157.2.7 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_SetLeapSecondsCmd_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

- Platform 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 485 of file default_cfe_time_fcncodes.h.

12.157.2.8 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_SetMETCmd_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)
- Platform 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 413 of file default_cfe_time_fcncodes.h.

12.157.2.9 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_SetSignalCmd_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 691 of file default_cfe_time_fcncodes.h.

12.157.2.10 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_SetSourceCmd_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_SourceSelect_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 195 of file default_cfe_time_fcncodes.h.

12.157.2.11 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_SetStateCmd_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" 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 \leftarrow 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 252 of file default_cfe_time_fcncodes.h.

12.157.2.12 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_SetSTCFCmd_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)
- Platform 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 450 of file default_cfe_time_fcncodes.h.

12.157.2.13 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) = ((current MET) + STCF) - (Leap Seconds)**

Command Mnemonic(s) \$sc_\$cpu_TIME_SetClock**Command Structure**[CFE_TIME_SetTimeCmd_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 subseconds 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)
- Platform 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 373 of file default_cfe_time_fcncodes.h.

12.157.2.14 CFE_TIME_SUB_ADJUST_CC #define CFE_TIME_SUB_ADJUST_CC 12 /* subtract one time S↔ TCF 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_SubAdjustCmd_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)
- Platform 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_ONE_HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC](#)

Definition at line 555 of file default_cfe_time_fcncodes.h.

12.157.2.15 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_SubDelayCmd_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)
- Platform 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 328 of file default_cfe_time_fcncodes.h.

12.157.2.16 CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC #define CFE_TIME_SUB_ONE_HZ_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_SubOneHzAdjustmentCmd_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_ONEHZ_EID](#) informational event message will be generated

Error Conditions

- Platform 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_ONEHZ_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_ONE_HZ_ADJUSTMENT_CC](#)

Definition at line 649 of file default_cfe_time_fcnccodes.h.

12.158 cfe/modules/time/config/default_cfe_time_interface_cfg.h File Reference

Macros

- #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 37
- #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_EPOCH_MICROS 0
- #define CFE_MISSION_TIME_FS_FACTOR 789004800

12.158.1 Detailed Description

CFE Time Services (CFE_TIME) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.158.2 Macro Definition Documentation

12.158.2.1 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 88 of file default_cfe_time_interface_cfg.h.

12.158.2.2 CFE_MISSION_TIME_AT_TONE_WILL_BE #define CFE_MISSION_TIME_AT_TONE_WILL_BE false
Definition at line 89 of file default_cfe_time_interface_cfg.h.

12.158.2.3 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 52 of file default_cfe_time_interface_cfg.h.

12.158.2.4 CFE_MISSION_TIME_CFG_DEFAULT_UTC #define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
Definition at line 53 of file default_cfe_time_interface_cfg.h.

12.158.2.5 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 65 of file default_cfe_time_interface_cfg.h.

12.158.2.6 CFE_MISSION_TIME_DEF_DELAY_SECS #define CFE_MISSION_TIME_DEF_DELAY_SECS 0
Definition at line 147 of file default_cfe_time_interface_cfg.h.

12.158.2.7 CFE_MISSION_TIME_DEF_DELAY_SUBS #define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
Definition at line 148 of file default_cfe_time_interface_cfg.h.

12.158.2.8 CFE_MISSION_TIME_DEF_LEAPS #define CFE_MISSION_TIME_DEF_LEAPS 37
Definition at line 145 of file default_cfe_time_interface_cfg.h.

12.158.2.9 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 ($\text{time} = \text{MET} + \text{STCF} - \text{Leap Seconds}$) 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 139 of file default_cfe_time_interface_cfg.h.

12.158.2.10 CFE_MISSION_TIME_DEF_MET_SUBS #define CFE_MISSION_TIME_DEF_MET_SUBS 0
Definition at line 140 of file default_cfe_time_interface_cfg.h.

12.158.2.11 CFE_MISSION_TIME_DEF_STCF_SECS #define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
Definition at line 142 of file default_cfe_time_interface_cfg.h.

12.158.2.12 CFE_MISSION_TIME_DEF_STCF_SUBS #define CFE_MISSION_TIME_DEF_STCF_SUBS 0
Definition at line 143 of file default_cfe_time_interface_cfg.h.

12.158.2.13 CFE_MISSION_TIME_EPOCH_DAY #define CFE_MISSION_TIME_EPOCH_DAY 1
Definition at line 166 of file default_cfe_time_interface_cfg.h.

12.158.2.14 CFE_MISSION_TIME_EPOCH_HOUR #define CFE_MISSION_TIME_EPOCH_HOUR 0
Definition at line 167 of file default_cfe_time_interface_cfg.h.

12.158.2.15 CFE_MISSION_TIME_EPOCH_MICROS #define CFE_MISSION_TIME_EPOCH_MICROS 0
Definition at line 170 of file default_cfe_time_interface_cfg.h.

12.158.2.16 CFE_MISSION_TIME_EPOCH_MINUTE #define CFE_MISSION_TIME_EPOCH_MINUTE 0
Definition at line 168 of file default_cfe_time_interface_cfg.h.

12.158.2.17 CFE_MISSION_TIME_EPOCH_SECOND #define CFE_MISSION_TIME_EPOCH_SECOND 0
Definition at line 169 of file default_cfe_time_interface_cfg.h.

12.158.2.18 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
Micros - 0 to 999999

Definition at line 165 of file default_cfe_time_interface_cfg.h.

12.158.2.19 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 208 of file default_cfe_time_interface_cfg.h.

12.158.2.20 CFE_MISSION_TIME_MAX_ELAPSED #define CFE_MISSION_TIME_MAX_ELAPSED 200000

Definition at line 114 of file default_cfe_time_interface_cfg.h.

12.158.2.21 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 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 113 of file default_cfe_time_interface_cfg.h.

12.159 cfe/modules/time/config/default_cfe_time_internal_cfg.h File Reference

Macros

- #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_TIME_START_TASK_PRIORITY 60
- #define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
- #define CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE 8192

12.159.1 Detailed Description

CFE Time Service (CFE_TIME) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.159.2 Macro Definition Documentation

12.159.2.1 CFE_PLATFORM_TIME_CFG_CLIENT #define CFE_PLATFORM_TIME_CFG_CLIENT false
Definition at line 48 of file default_cfe_time_internal_cfg.h.

12.159.2.2 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 dictates the period at which the simulated 'last tone' time is updated. Units are seconds.

Limits

Not Applicable

Definition at line 205 of file default_cfe_time_internal_cfg.h.

12.159.2.3 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_TIME_CFG_CLIENT AS true. The other must be defined as false.

Definition at line 47 of file default_cfe_time_internal_cfg.h.

12.159.2.4 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 definition 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_CFG_SIGNAL define to true to enable tone signal commands.

Limits

Not Applicable

Definition at line 95 of file default_cfe_time_internal_cfg.h.

12.159.2.5 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 115 of file [default_cfe_time_internal_cfg.h](#).

12.159.2.6 CFE_PLATFORM_TIME_CFG_SRC_GPS #define CFE_PLATFORM_TIME_CFG_SRC_GPS false

Definition at line 132 of file [default_cfe_time_internal_cfg.h](#).

12.159.2.7 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_CFG_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_TIME_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 131 of file [default_cfe_time_internal_cfg.h](#).

12.159.2.8 CFE_PLATFORM_TIME_CFG_SRC_TIME #define CFE_PLATFORM_TIME_CFG_SRC_TIME false

Definition at line 133 of file [default_cfe_time_internal_cfg.h](#).

12.159.2.9 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 192 of file default_cfe_time_internal_cfg.h.

12.159.2.10 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 180 of file default_cfe_time_internal_cfg.h.

12.159.2.11 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 80 of file default_cfe_time_internal_cfg.h.

12.159.2.12 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 152 of file default_cfe_time_internal_cfg.h.

12.159.2.13 CFE_PLATFORM_TIME_MAX_DELTA_SUBS #define CFE_PLATFORM_TIME_MAX_DELTA_SU←
BS 500000

Definition at line 153 of file default_cfe_time_internal_cfg.h.

12.159.2.14 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 165 of file default_cfe_time_internal_cfg.h.

12.159.2.15 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS #define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
Definition at line 166 of file default_cfe_time_internal_cfg.h.**12.159.2.16 CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY** #define CFE_PLATFORM_TIME_ONEHZ_TASK_P←
RIORITY 25
Definition at line 222 of file default_cfe_time_internal_cfg.h.**12.159.2.17 CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE** #define CFE_PLATFORM_TIME_ONEHZ_TASK←
_STACK_SIZE 8192
Definition at line 241 of file default_cfe_time_internal_cfg.h.

12.159.2.18 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 parameters. Remember that the meaning of each task priority is inverted – a "lower" number has a "higher" priority.

Definition at line 220 of file default_cfe_time_internal_cfg.h.

12.159.2.19 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 parameters. There are no restrictions on the upper limit however, the maximum stack 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 239 of file default_cfe_time_internal_cfg.h.

12.159.2.20 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY #define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25

Definition at line 221 of file default_cfe_time_internal_cfg.h.

12.159.2.21 CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096

Definition at line 240 of file default_cfe_time_internal_cfg.h.

12.160 cfe/modules/time/config/default_cfe_time_mission_cfg.h File Reference

```
#include "cfe_time_interface_cfg.h"
```

12.160.1 Detailed Description

CFE Time Services (CFE_TIME) Application Mission Configuration Header File

This is a compatibility header for the "mission_cfg.h" file that has traditionally provided public config definitions for each CFS app.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.161 cfe/modules/time/config/default_cfe_time_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_time_fcncodes.h"
#include "cfe_time_msgdefs.h"
#include "cfe_time_msgstruct.h"
```

12.161.1 Detailed Description

Specification for the CFE Time Services (CFE_TIME) command and telemetry message data types.

This is a compatibility header for the "cfe_time_msg.h" file that has traditionally provided the message definitions for cFS apps.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.162 cfe/modules/time/config/default_cfe_time_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_time_extern_typedefs.h"
#include "cfe_time_fcncodes.h"
```

Data Structures

- struct [CFE_TIME_LeapsCmd_Payload](#)
Set leap seconds command payload.
- struct [CFE_TIME_StateCmd_Payload](#)
Set clock state command payload.
- struct [CFE_TIME_SourceCmd_Payload](#)
Set time data source command payload.
- struct [CFE_TIME_SignalCmd_Payload](#)
Set tone signal source command payload.
- struct [CFE_TIME_TimeCmd_Payload](#)
Generic seconds, microseconds command payload.
- struct [CFE_TIME_OneHzAdjustmentCmd_Payload](#)
Generic seconds, subseconds command payload.
- struct [CFE_TIME_ToneDataCmd_Payload](#)
Time at tone data command payload.
- struct [CFE_TIME_HousekeepingTlm_Payload](#)
- struct [CFE_TIME_DiagnosticTlm_Payload](#)

Macros

- #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_REFERR 0x0010
GetReference read error, will be set if unable to get a consistent ref value.
- #define CFE_TIME_FLAG_UNUSED 0x000F
Reserved flags - should be zero.

Typedefs

- typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t
Set leap seconds command payload.
- typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t
Set clock state command payload.
- typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t
Set time data source command payload.
- typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t
Set tone signal source command payload.
- typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t
Generic seconds, microseconds command payload.
- typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t
Generic seconds, subseconds command payload.
- typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t
Time at tone data command payload.
- typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t
- typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t

12.162.1 Detailed Description

Specification for the CFE Time Services (CFE_TIME) command and telemetry message constant definitions.
For CFE_TIME this is only the function/command code definitions

12.162.2 Typedef Documentation

12.162.2.1 **CFE_TIME_DiagnosticTlm_Payload_t** `typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t`

Name Time Services Diagnostics Packet

12.162.2.2 **CFE_TIME_HousekeepingTlm_Payload_t** `typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t`

Name Time Services Housekeeping Packet

12.162.2.3 **CFE_TIME_LeapsCmd_Payload_t** `typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t` Set leap seconds command payload.

12.162.2.4 **CFE_TIME_OneHzAdjustmentCmd_Payload_t** `typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t` Generic seconds, subseconds command payload.

12.162.2.5 **CFE_TIME_SignalCmd_Payload_t** `typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t` Set tone signal source command payload.

12.162.2.6 **CFE_TIME_SourceCmd_Payload_t** `typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t` Set time data source command payload.

12.162.2.7 **CFE_TIME_StateCmd_Payload_t** `typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t` Set clock state command payload.

12.162.2.8 **CFE_TIME_TimeCmd_Payload_t** `typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t` Generic seconds, microseconds command payload.

12.162.2.9 **CFE_TIME_ToneDataCmd_Payload_t** `typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t` Time at tone data command payload.

12.163 cfe/modules/time/config/default_cfe_time_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_time_topicids.h"
```

Macros

- #define CFE_TIME_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_CMD_TOPICID)
/* 0x1805 */
- #define CFE_TIME_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_SEND_HK_TOPICID)
/* 0x180D */
- #define CFE_TIME_TONE_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_TONE_CMD_TOPICID)
/* 0x1810 */
- #define CFE_TIME_ONEHZ_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_ONEHZ_CMD_TOPICID)
/* 0x1811 */
- #define CFE_TIME_DATA_CMD_MID CFE_GLOBAL_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_DATA_CMD_TOPICID)
/* 0x1860 */
- #define CFE_TIME_SEND_CMD_MID CFE_GLOBAL_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_SEND_CMD_TOPICID)
/* 0x1862 */
- #define CFE_TIME_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TIME_HK_TLM_TOPICID)
/* 0x0805 */
- #define CFE_TIME_DIAG_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TIME_DIAG_TLM_TOPICID)
/* 0x0806 */
- #define CFE_TIME_1HZ_CMD_MID CFE_TIME_ONEHZ_CMD_MID

12.163.1 Detailed Description

CFE Time Services (CFE_TIME) Application Message IDs

12.163.2 Macro Definition Documentation

12.163.2.1 CFE_TIME_1HZ_CMD_MID #define CFE_TIME_1HZ_CMD_MID CFE_TIME_ONEHZ_CMD_MID
Definition at line 55 of file default_cfe_time_msgids.h.

12.163.2.2 CFE_TIME_CMD_MID #define CFE_TIME_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_CMD_TOPICID)
/* 0x1805 */
Definition at line 32 of file default_cfe_time_msgids.h.

12.163.2.3 CFE_TIME_DATA_CMD_MID #define CFE_TIME_DATA_CMD_MID CFE_GLOBAL_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_DATA_CMD_TOPICID)
/* 0x1860 */
Definition at line 40 of file default_cfe_time_msgids.h.

12.163.2.4 CFE_TIME_DIAG_TLM_MID #define CFE_TIME_DIAG_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TIME_DIAG_TLM_TOPICID)
/* 0x0806 */
Definition at line 47 of file default_cfe_time_msgids.h.

12.163.2.5 CFE_TIME_HK_TLM_MID #define CFE_TIME_HK_TLM_MID CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TIME_HK_TLM_TOPICID)
/* 0x0805 */
Definition at line 46 of file default_cfe_time_msgids.h.

12.163.2.6 CFE_TIME_ONEHZ_CMD_MID #define CFE_TIME_ONEHZ_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_MID)
/* 0x1811 */
Definition at line 35 of file default_cfe_time_msgids.h.

12.163.2.7 CFE_TIME_SEND_CMD_MID #define CFE_TIME_SEND_CMD_MID CFE_GLOBAL_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_MID)
/* 0x1862 */
Definition at line 41 of file default_cfe_time_msgids.h.

12.163.2.8 CFE_TIME_SEND_HK_MID #define CFE_TIME_SEND_HK_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_MID)
/* 0x180D */
Definition at line 33 of file default_cfe_time_msgids.h.

12.163.2.9 CFE_TIME_TONE_CMD_MID #define CFE_TIME_TONE_CMD_MID CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_MID)
/* 0x1810 */
Definition at line 34 of file default_cfe_time_msgids.h.

12.164 cfe/modules/time/config/default_cfe_time_msgstruct.h File Reference

```
#include "cfe_time_msgdefs.h"
#include "cfe_msg_hdr.h"
```

Data Structures

- struct [CFE_TIME_NoopCmd](#)
- struct [CFE_TIME_ResetCountersCmd](#)
- struct [CFE_TIME_SendDiagnosticCmd](#)
- struct [CFE_TIME_OneHzCmd](#)
- struct [CFE_TIME_ToneSignalCmd](#)
- struct [CFE_TIME_FakeToneCmd](#)
- struct [CFE_TIME_SendHkCmd](#)
- struct [CFE_TIME_SetLeapSecondsCmd](#)
Set leap seconds command.
- struct [CFE_TIME_SetStateCmd](#)
Set clock state command.
- struct [CFE_TIME_SetSourceCmd](#)
Set time data source command.
- struct [CFE_TIME_SetSignalCmd](#)
Set tone signal source command.
- struct [CFE_TIME_AddDelayCmd](#)
- struct [CFE_TIME_SubDelayCmd](#)
- struct [CFE_TIME_SetMETCCmd](#)
- struct [CFE_TIME_SetSTCFCmd](#)
- struct [CFE_TIME_AddAdjustCmd](#)
- struct [CFE_TIME_SubAdjustCmd](#)
- struct [CFE_TIME_SetTimeCmd](#)
- struct [CFE_TIME_AddOneHzAdjustmentCmd](#)
- struct [CFE_TIME_SubOneHzAdjustmentCmd](#)

- struct [CFE_TIME_ToneDataCmd](#)
Time at tone data command.
- struct [CFE_TIME_HousekeepingTlm](#)
- struct [CFE_TIME_DiagnosticTlm](#)

Typedefs

- typedef struct [CFE_TIME_NoopCmd](#) CFE_TIME_NoopCmd_t
- typedef struct [CFE_TIME_ResetCountersCmd](#) CFE_TIME_ResetCountersCmd_t
- typedef struct [CFE_TIME_SendDiagnosticCmd](#) CFE_TIME_SendDiagnosticCmd_t
- typedef struct [CFE_TIME_OneHzCmd](#) CFE_TIME_OneHzCmd_t
- typedef struct [CFE_TIME_ToneSignalCmd](#) CFE_TIME_ToneSignalCmd_t
- typedef struct [CFE_TIME_FakeToneCmd](#) CFE_TIME_FakeToneCmd_t
- typedef struct [CFE_TIME_SendHkCmd](#) CFE_TIME_SendHkCmd_t
- typedef struct [CFE_TIME_SetLeapSecondsCmd](#) CFE_TIME_SetLeapSecondsCmd_t
Set leap seconds command.
- typedef struct [CFE_TIME_SetStateCmd](#) CFE_TIME_SetStateCmd_t
Set clock state command.
- typedef struct [CFE_TIME_SetSourceCmd](#) CFE_TIME_SetSourceCmd_t
Set time data source command.
- typedef struct [CFE_TIME_SetSignalCmd](#) CFE_TIME_SetSignalCmd_t
Set tone signal source command.
- typedef struct [CFE_TIME_AddDelayCmd](#) CFE_TIME_AddDelayCmd_t
- typedef struct [CFE_TIME_SubDelayCmd](#) CFE_TIME_SubDelayCmd_t
- typedef struct [CFE_TIME_SetMETCmd](#) CFE_TIME_SetMETCmd_t
- typedef struct [CFE_TIME_SetSTCFCmd](#) CFE_TIME_SetSTCFCmd_t
- typedef struct [CFE_TIME_AddAdjustCmd](#) CFE_TIME_AddAdjustCmd_t
- typedef struct [CFE_TIME_SubAdjustCmd](#) CFE_TIME_SubAdjustCmd_t
- typedef struct [CFE_TIME_SetTimeCmd](#) CFE_TIME_SetTimeCmd_t
- typedef struct [CFE_TIME_AddOneHzAdjustmentCmd](#) CFE_TIME_AddOneHzAdjustmentCmd_t
- typedef struct [CFE_TIME_SubOneHzAdjustmentCmd](#) CFE_TIME_SubOneHzAdjustmentCmd_t
- typedef struct [CFE_TIME_ToneDataCmd](#) CFE_TIME_ToneDataCmd_t
Time at tone data command.
- typedef struct [CFE_TIME_HousekeepingTlm](#) CFE_TIME_HousekeepingTlm_t
- typedef struct [CFE_TIME_DiagnosticTlm](#) CFE_TIME_DiagnosticTlm_t

12.164.1 Detailed Description

Purpose: cFE Executive Services (TIME) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide
Notes:

12.164.2 Typedef Documentation

12.164.2.1 CFE_TIME_AddAdjustCmd_t [typedef struct CFE_TIME_AddAdjustCmd](#) CFE_TIME_AddAdjustCmd_t

12.164.2.2 CFE_TIME_AddDelayCmd_t [typedef struct CFE_TIME_AddDelayCmd](#) CFE_TIME_AddDelayCmd_t

12.164.2.3 CFE_TIME_AddOneHzAdjustmentCmd_t `typedef struct CFE_TIME_AddOneHzAdjustmentCmd CFE_TIME_AddOneHzAdjustmentCmd_t`

12.164.2.4 CFE_TIME_DiagnosticTlm_t `typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t`

12.164.2.5 CFE_TIME_FakeToneCmd_t `typedef struct CFE_TIME_FakeToneCmd CFE_TIME_FakeToneCmd_t`

12.164.2.6 CFE_TIME_HousekeepingTlm_t `typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t`

12.164.2.7 CFE_TIME_NoopCmd_t `typedef struct CFE_TIME_NoopCmd CFE_TIME_NoopCmd_t`

12.164.2.8 CFE_TIME_OneHzCmd_t `typedef struct CFE_TIME_OneHzCmd CFE_TIME_OneHzCmd_t`

12.164.2.9 CFE_TIME_ResetCountersCmd_t `typedef struct CFE_TIME_ResetCountersCmd CFE_TIME_ResetCountersCmd_t`

12.164.2.10 CFE_TIME_SendDiagnosticCmd_t `typedef struct CFE_TIME_SendDiagnosticCmd CFE_TIME_SendDiagnosticCmd_t`

12.164.2.11 CFE_TIME_SendHkCmd_t `typedef struct CFE_TIME_SendHkCmd CFE_TIME_SendHkCmd_t`

12.164.2.12 CFE_TIME_SetLeapSecondsCmd_t `typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t`
Set leap seconds command.

12.164.2.13 CFE_TIME_SetMETCmd_t `typedef struct CFE_TIME_SetMETCmd CFE_TIME_SetMETCmd_t`

12.164.2.14 CFE_TIME_SetSignalCmd_t `typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t`
Set tone signal source command.

12.164.2.15 CFE_TIME_SetSourceCmd_t `typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t`
Set time data source command.

12.164.2.16 CFE_TIME_SetStateCmd_t `typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t`
Set clock state command.

12.164.2.17 CFE_TIME_SetSTCFCmd_t `typedef struct CFE_TIME_SetSTCFCmd CFE_TIME_SetSTCFCmd_t`

12.164.2.18 CFE_TIME_SetTimeCmd_t `typedef struct CFE_TIME_SetTimeCmd CFE_TIME_SetTimeCmd_t`

12.164.2.19 CFE_TIME_SubAdjustCmd_t `typedef struct CFE_TIME_SubAdjustCmd CFE_TIME_SubAdjustCmd_t`

12.164.2.20 CFE_TIME_SubDelayCmd_t `typedef struct CFE_TIME_SubDelayCmd CFE_TIME_SubDelayCmd_t`

12.164.2.21 CFE_TIME_SubOneHzAdjustmentCmd_t `typedef struct CFE_TIME_SubOneHzAdjustmentCmd CFE_TIME_SubOneHzAdjustmentCmd_t`

12.164.2.22 CFE_TIME_ToneDataCmd_t `typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t`
Time at tone data command.

12.164.2.23 CFE_TIME_ToneSignalCmd_t `typedef struct CFE_TIME_ToneSignalCmd CFE_TIME_ToneSignalCmd_t`

12.165 cfe/modules/time/config/default_cfe_time_platform_cfg.h File Reference

```
#include "cfe_time_mission_cfg.h"
#include "cfe_time_internal_cfg.h"
```

12.165.1 Detailed Description

CFE Time Services (CFE_TIME) Application Platform Configuration Header File

This is a compatibility header for the "platform_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

12.166 cfe/modules/time/config/default_cfe_time_topicids.h File Reference

Macros

- `#define CFE_MISSION_TIME_CMD_TOPICID 5`
- `#define CFE_MISSION_TIME_SEND_HK_TOPICID 13`
- `#define CFE_MISSION_TIME_TONE_CMD_TOPICID 16`
- `#define CFE_MISSION_TIME_ONEHZ_CMD_TOPICID 17`
- `#define CFE_MISSION_TIME_DATA_CMD_TOPICID 0`
- `#define CFE_MISSION_TIME_SEND_CMD_TOPICID 2`
- `#define CFE_MISSION_TIME_HK_TLM_TOPICID 5`
- `#define CFE_MISSION_TIME_DIAG_TLM_TOPICID 6`

12.166.1 Detailed Description

CFE Time Services (CFE_TIME) Application Topic IDs

12.166.2 Macro Definition Documentation

12.166.2.1 CFE_MISSION_TIME_CMD_TOPICID #define CFE_MISSION_TIME_CMD_TOPICID 5

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE command messages

Limits

Not Applicable

Definition at line 35 of file default_cfe_time_topicids.h.

12.166.2.2 CFE_MISSION_TIME_DATA_CMD_TOPICID #define CFE_MISSION_TIME_DATA_CMD_TOPICID 0

Purpose cFE Portable Message Numbers for Global Messages

Description:

Portable message numbers for the cFE global messages

Limits

Not Applicable

Definition at line 49 of file default_cfe_time_topicids.h.

12.166.2.3 CFE_MISSION_TIME_DIAG_TLM_TOPICID #define CFE_MISSION_TIME_DIAG_TLM_TOPICID 6

Definition at line 62 of file default_cfe_time_topicids.h.

12.166.2.4 CFE_MISSION_TIME_HK_TLM_TOPICID #define CFE_MISSION_TIME_HK_TLM_TOPICID 5

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE telemetry messages

Limits

Not Applicable

Definition at line 61 of file default_cfe_time_topicids.h.

12.166.2.5 CFE_MISSION_TIME_ONEHZ_CMD_TOPICID #define CFE_MISSION_TIME_ONEHZ_CMD_TOPICID 17

Definition at line 38 of file default_cfe_time_topicids.h.

12.166.2.6 CFE_MISSION_TIME_SEND_CMD_TOPICID #define CFE_MISSION_TIME_SEND_CMD_TOPICID 2
Definition at line 50 of file default_cfe_time_topicids.h.

12.166.2.7 CFE_MISSION_TIME_SEND_HK_TOPICID #define CFE_MISSION_TIME_SEND_HK_TOPICID 13
Definition at line 36 of file default_cfe_time_topicids.h.

12.166.2.8 CFE_MISSION_TIME_TONE_CMD_TOPICID #define CFE_MISSION_TIME_TONE_CMD_TOPICID 16
Definition at line 37 of file default_cfe_time_topicids.h.

12.167 cfe/modules/time/fsw/inc/cfe_time_eventids.h File Reference

Macros

TIME event IDs

- #define [CFE_TIME_INIT_EID](#) 1
TIME Initialization Event ID.
- #define [CFE_TIME_NOOP_EID](#) 4
TIME No-op Command Success Event ID.
- #define [CFE_TIME_RESET_EID](#) 5
TIME Reset Counters Command Success Event ID.
- #define [CFE_TIME_DIAG_EID](#) 6
TIME Request Diagnostics Command Success Event ID.
- #define [CFE_TIME_STATE_EID](#) 7
TIME Set Time State Command Success Event ID.
- #define [CFE_TIME_SOURCE_EID](#) 8
TIME Set Time Source Command Success Event ID.
- #define [CFE_TIME_SIGNAL_EID](#) 9
TIME Set Tone Source Command Success Event ID.
- #define [CFE_TIME_DELAY_EID](#) 11
TIME Add or Subtract Delay Command Success Event ID.
- #define [CFE_TIME_TIME_EID](#) 12
TIME Set Time Command Success Event ID.
- #define [CFE_TIME_MET_EID](#) 13
TIME Set Mission Elapsed Time Command Success Event ID.
- #define [CFE_TIME_STCF_EID](#) 14
TIME Set Spacecraft Time Correlation Factor Command Success Event ID.
- #define [CFE_TIME_DELTA_EID](#) 15
TIME Add or Subtract Single STCF Adjustment Command Success Event ID.
- #define [CFE_TIME_ONEHZ_EID](#) 16
TIME Add or Subtract STCF Adjustment Each Second Command Success Event ID.
- #define [CFE_TIME_LEAPS_EID](#) 17
TIME Set Leap Seconds Command Success Event ID.
- #define [CFE_TIME_FLY_ON_EID](#) 20
TIME Entered FLYWHEEL Mode Event ID.
- #define [CFE_TIME_FLY_OFF_EID](#) 21
TIME Exited FLYWHEEL Mode Event ID.
- #define [CFE_TIME_ID_ERR_EID](#) 26
TIME Invalid Message ID Received Event ID.
- #define [CFE_TIME_CC_ERR_EID](#) 27
TIME Invalid Command Code Received Event ID.
- #define [CFE_TIME_STATE_ERR_EID](#) 30
TIME Set Clock State Command Invalid State Event ID.

- #define `CFE_TIME_SOURCE_ERR_EID` 31
TIME Set Clock Source Command Invalid Source Event ID.
- #define `CFE_TIME_SIGNAL_ERR_EID` 32
TIME Set Clock Tone Source Command Invalid Source Event ID.
- #define `CFE_TIME_DELAY_ERR_EID` 33
TIME Add or Subtract Tone Delay Command Invalid Time Value Event ID.
- #define `CFE_TIME_TIME_ERR_EID` 34
TIME Set Spacecraft Time Command Invalid Time Value Event ID.
- #define `CFE_TIME_MET_ERR_EID` 35
TIME Set Mission Elapsed Time Command Invalid Time Value Event ID.
- #define `CFE_TIME_STCF_ERR_EID` 36
TIME Set Spacecraft Time Correlation Factor Command Invalid Time Value Event ID.
- #define `CFE_TIME_DELTA_ERR_EID` 37
TIME Add or Subtract Single STCF Adjustment Command Invalid Time Value Event ID.
- #define `CFE_TIME_SOURCE_CFG_EID` 40
TIME Set Clock Source Command Incompatible Mode Event ID.
- #define `CFE_TIME_SIGNAL_CFG_EID` 41
TIME Set Clock Signal Command Incompatible Mode Event ID.
- #define `CFE_TIME_DELAY_CFG_EID` 42
TIME Add or Subtract Tone Delay Command Incompatible Mode Event ID.
- #define `CFE_TIME_TIME_CFG_EID` 43
TIME Set Spacecraft Time Command Incompatible Mode Event ID.
- #define `CFE_TIME_MET_CFG_EID` 44
TIME Set Mission Elapsed Time Command Incompatible Mode Event ID.
- #define `CFE_TIME_STCF_CFG_EID` 45
TIME Set Spacecraft Time Correlation Factor Command Incompatible Mode Event ID.
- #define `CFE_TIME_LEAPS_CFG_EID` 46
TIME Set Leap Seconds Command Incompatible Mode Event ID.
- #define `CFE_TIME_DELTA_CFG_EID` 47
TIME Add or Subtract Single STCF Adjustment Command Incompatible Mode Event ID.
- #define `CFE_TIME_ONEHZ_CFG_EID` 48
TIME Add or Subtract STCF Adjustment Each Second Command Incompatible Mode Event ID.
- #define `CFE_TIME_LEN_ERR_EID` 49
TIME Invalid Command Length Event ID.

12.167.1 Detailed Description

cFE Time Services Event IDs

12.167.2 Macro Definition Documentation

12.167.2.1 CFE_TIME_CC_ERR_EID #define `CFE_TIME_CC_ERR_EID` 27
TIME Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID `CFE_TIME_CMD_MID` received on the TIME message pipe.
Definition at line 232 of file cfe_time_eventids.h.

12.167.2.2 CFE_TIME_DELAY_CFG_EID #define CFE_TIME_DELAY_CFG_EID 42
TIME Add or Subtract Tone Delay Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add Tone Delay Command](#) OR [TIME Subtract Tone Delay Command](#) failure due to being in an incompatible mode.

Definition at line 364 of file cfe_time_eventids.h.

12.167.2.3 CFE_TIME_DELAY_EID #define CFE_TIME_DELAY_EID 11
TIME Add or Subtract Delay Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Add Time Delay Command](#) OR a [Subtract Time Delay Command](#) success.

Definition at line 120 of file cfe_time_eventids.h.

12.167.2.4 CFE_TIME_DELAY_ERR_EID #define CFE_TIME_DELAY_ERR_EID 33
TIME Add or Subtract Tone Delay Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Add Tone Delay Command](#) OR [TIME Subtract Tone Delay Command](#) failure due to an invalid time value.

Definition at line 278 of file cfe_time_eventids.h.

12.167.2.5 CFE_TIME_DELTA_CFG_EID #define CFE_TIME_DELTA_CFG_EID 47
TIME Add or Subtract Single STCF Adjustment Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add Single STCF Adjustment Command](#) OR [TIME Subtract Single STCF Adjustment Command](#) failure due to being in an incompatible mode.

Definition at line 425 of file cfe_time_eventids.h.

12.167.2.6 CFE_TIME_DELTA_EID #define CFE_TIME_DELTA_EID 15
TIME Add or Subtract Single STCF Adjustment Command Success Event ID.

Type: INFORMATION

Cause:

TIME Add Single STCF Adjustment Command OR TIME Subtract Single STCF Adjustment Command success.
Definition at line 165 of file cfe_time_eventids.h.

12.167.2.7 CFE_TIME_DELTA_ERR_EID #define CFE_TIME_DELTA_ERR_EID 37
TIME Add or Subtract Single STCF Adjustment Command Invalid Time Value Event ID.

Type: ERROR

Cause:

TIME Add Single STCF Adjustment Command OR TIME Subtract Single STCF Adjustment Command failure due to
an invalid time value.
Definition at line 327 of file cfe_time_eventids.h.

12.167.2.8 CFE_TIME_DIAG_EID #define CFE_TIME_DIAG_EID 6
TIME Request Diagnostics Command Success Event ID.

Type: DEBUG

Cause:

TIME Request Diagnostics Command success.
Definition at line 75 of file cfe_time_eventids.h.

12.167.2.9 CFE_TIME_FLY_OFF_EID #define CFE_TIME_FLY_OFF_EID 21
TIME Exited FLYWHEEL Mode Event ID.

Type: INFORMATION

Cause:

TIME Exited FLYWHEEL Mode.
Definition at line 210 of file cfe_time_eventids.h.

12.167.2.10 CFE_TIME_FLY_ON_EID #define CFE_TIME_FLY_ON_EID 20
TIME Entered FLYWHEEL Mode Event ID.

Type: INFORMATION

Cause:

TIME Entered FLYWHEEL Mode.
Definition at line 199 of file cfe_time_eventids.h.

12.167.2.11 CFE_TIME_ID_ERR_EID #define CFE_TIME_ID_ERR_EID 26
TIME Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the TIME message pipe.
Definition at line 221 of file cfe_time_eventids.h.

12.167.2.12 CFE_TIME_INIT_EID #define CFE_TIME_INIT_EID 1
TIME Initialization Event ID.

Type: INFORMATION

Cause:

Time Services Task Initialization complete.
Definition at line 42 of file cfe_time_eventids.h.

12.167.2.13 CFE_TIME_LEAPS_CFG_EID #define CFE_TIME_LEAPS_CFG_EID 46
TIME Set Leap Seconds Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Leap Seconds Command](#) failure due to being in an incompatible mode.
Definition at line 412 of file cfe_time_eventids.h.

12.167.2.14 CFE_TIME_LEAPS_EID #define CFE_TIME_LEAPS_EID 17
TIME Set Leap Seconds Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Leap Seconds Command](#) success.
Definition at line 188 of file cfe_time_eventids.h.

12.167.2.15 CFE_TIME_LEN_ERR_EID #define CFE_TIME_LEN_ERR_EID 49
TIME Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE_TIME_CMD_MID](#) received on the TIME message pipe.
Definition at line 450 of file cfe_time_eventids.h.

12.167.2.16 CFE_TIME_MET_CFG_EID #define CFE_TIME_MET_CFG_EID 44
TIME Set Mission Elapsed Time Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Mission Elapsed Time Command](#) failure due to being in an incompatible mode.
Definition at line 388 of file cfe_time_eventids.h.

12.167.2.17 CFE_TIME_MET_EID #define CFE_TIME_MET_EID 13
TIME Set Mission Elapsed Time Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Mission Elapsed Time Command](#) success.
Definition at line 142 of file cfe_time_eventids.h.

12.167.2.18 CFE_TIME_MET_ERR_EID #define CFE_TIME_MET_ERR_EID 35
TIME Set Mission Elapsed Time Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Set Mission Elapsed Time Command](#) failure due to an invalid time value.
Definition at line 302 of file cfe_time_eventids.h.

12.167.2.19 CFE_TIME_NOOP_EID #define CFE_TIME_NOOP_EID 4
TIME No-op Command Success Event ID.

Type: INFORMATION

Cause:

[TIME NO-OP Command](#) success.
Definition at line 53 of file cfe_time_eventids.h.

12.167.2.20 CFE_TIME_ONEHZ_CFG_EID #define CFE_TIME_ONEHZ_CFG_EID 48
TIME Add or Subtract STCF Adjustment Each Second Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add STCF Adjustment Each Second Command](#) OR [TIME Subtract STCF Adjustment Each Second Command](#) failure due to being in an incompatible mode.
Definition at line 438 of file cfe_time_eventids.h.

12.167.2.21 CFE_TIME_ONEHZ_EID #define CFE_TIME_ONEHZ_EID 16
TIME Add or Subtract STCF Adjustment Each Second Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Add STCF Adjustment Each Second Command](#) OR [TIME Subtract STCF Adjustment Each Second Command](#) success.
Definition at line 177 of file cfe_time_eventids.h.

12.167.2.22 CFE_TIME_RESET_EID #define CFE_TIME_RESET_EID 5
TIME Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[TIME Reset Counters Command](#) success.
Definition at line 64 of file cfe_time_eventids.h.

12.167.2.23 CFE_TIME_SIGNAL_CFG_EID #define CFE_TIME_SIGNAL_CFG_EID 41
TIME Set Clock Signal Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Clock Signal Command](#) failure due to being in an incompatible mode.
Definition at line 351 of file cfe_time_eventids.h.

12.167.2.24 CFE_TIME_SIGNAL_EID #define CFE_TIME_SIGNAL_EID 9
TIME Set Tone Source Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Clock Tone Source Command](#) success.
Definition at line 108 of file cfe_time_eventids.h.

12.167.2.25 CFE_TIME_SIGNAL_ERR_EID #define CFE_TIME_SIGNAL_ERR_EID 32
TIME Set Clock Tone Source Command Invalid Source Event ID.

Type: ERROR

Cause:

[Set Clock Tone Source Command](#) failed due to invalid source requested.
Definition at line 265 of file cfe_time_eventids.h.

12.167.2.26 CFE_TIME_SOURCE_CFG_EID #define CFE_TIME_SOURCE_CFG_EID 40
TIME Set Clock Source Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Clock Source Command](#) failure due to being in an incompatible mode.
Definition at line 339 of file cfe_time_eventids.h.

12.167.2.27 CFE_TIME_SOURCE_EID #define CFE_TIME_SOURCE_EID 8
TIME Set Time Source Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Time Source Command](#) success.
Definition at line 97 of file cfe_time_eventids.h.

12.167.2.28 CFE_TIME_SOURCE_ERR_EID #define CFE_TIME_SOURCE_ERR_EID 31
TIME Set Clock Source Command Invalid Source Event ID.

Type: ERROR

Cause:

[TIME Set Clock Source Command](#) failed due to invalid source requested.
Definition at line 254 of file cfe_time_eventids.h.

12.167.2.29 CFE_TIME_STATE_EID #define CFE_TIME_STATE_EID 7
TIME Set Time State Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Time State Command](#) success.
Definition at line 86 of file cfe_time_eventids.h.

12.167.2.30 CFE_TIME_STATE_ERR_EID #define CFE_TIME_STATE_ERR_EID 30
TIME Set Clock State Command Invalid State Event ID.

Type: ERROR

Cause:

[TIME Set Clock State Command](#) failed due to invalid state requested.
Definition at line 243 of file cfe_time_eventids.h.

12.167.2.31 CFE_TIME_STCF_CFG_EID #define CFE_TIME_STCF_CFG_EID 45
TIME Set Spacecraft Time Correlation Factor Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) failure due to being in an incompatible mode.
Definition at line 400 of file cfe_time_eventids.h.

12.167.2.32 CFE_TIME_STCF_EID #define CFE_TIME_STCF_EID 14
TIME Set Spacecraft Time Correlation Factor Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) success.
Definition at line 153 of file cfe_time_eventids.h.

12.167.2.33 CFE_TIME_STCF_ERR_EID #define CFE_TIME_STCF_ERR_EID 36
TIME Set Spacecraft Time Correlation Factor Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) failure due to an invalid time value.
Definition at line 314 of file cfe_time_eventids.h.

12.167.2.34 CFE_TIME_TIME_CFG_EID #define CFE_TIME_TIME_CFG_EID 43
TIME Set Spacecraft Time Command Incompatible Mode Event ID.

Type: ERROR

Cause:

TIME Set Spacecraft Time Command failure due to being in an incompatible mode.
Definition at line 376 of file cfe_time_eventids.h.

12.167.2.35 CFE_TIME_TIME_EID #define CFE_TIME_TIME_EID 12
TIME Set Time Command Success Event ID.

Type: INFORMATION

Cause:

TIME Set Time Command success.
Definition at line 131 of file cfe_time_eventids.h.

12.167.2.36 CFE_TIME_TIME_ERR_EID #define CFE_TIME_TIME_ERR_EID 34
TIME Set Spacecraft Time Command Invalid Time Value Event ID.

Type: ERROR

Cause:

TIME Set Spacecraft Time Command failure due to an invalid time value.
Definition at line 290 of file cfe_time_eventids.h.

12.168 osal/docs/src/osal_frontpage.dox File Reference

12.169 osal/docs/src/osal_fs.dox File Reference

12.170 osal/docs/src/osal_timer.dox File Reference

12.171 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_USED`
- #define `OS_PRINTF`(n, m)
- #define `OSAL_SIZE_C`(X) `((size_t)(X))`
- #define `OSAL_BLOCKCOUNT_C`(X) `((osal_blockcount_t)(X))`
- #define `OSAL_INDEX_C`(X) `((osal_index_t)(X))`
- #define `OSAL_OBJTYPE_C`(X) `((osal_objtype_t)(X))`
- #define `OSAL_STATUS_C`(X) `((osal_status_t)(X))`

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 uint32 osal_id_t`
- `typedef size_t osal_blockcount_t`
- `typedef uint32 osal_index_t`
- `typedef uint32 osal_objtype_t`
- `typedef int32 osal_status_t`
- `typedef void(* OS_ArgCallback_t) (osal_id_t object_id, void *arg)`

General purpose OSAL callback function.

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)`

12.171.1 Detailed Description

Purpose: Unit specification for common types.

Design Notes: Assumes make file has defined processor family

12.171.2 Macro Definition Documentation

12.171.2.1 `_EXTENSION_` #define _EXTENSION_

Definition at line 65 of file common_types.h.

12.171.2.2 `CompileTimeAssert` #define CompileTimeAssert(

Condition,

Message) typedef char Message[(*Condition*) ? 1 : -1]

Definition at line 48 of file common_types.h.

12.171.2.3 `OS_PRINTF` #define OS_PRINTF(

n,

m)

Definition at line 67 of file common_types.h.

12.171.2.4 `OS_USED` #define OS_USED

Definition at line 66 of file common_types.h.

12.171.2.5 `OSAL_BLOCKCOUNT_C` #define OSAL_BLOCKCOUNT_C(

X) ((*osal_blockcount_t*)(*X*))

Definition at line 172 of file common_types.h.

12.171.2.6 `OSAL_INDEX_C` #define OSAL_INDEX_C(

X) ((*osal_index_t*)(*X*))

Definition at line 173 of file common_types.h.

12.171.2.7 `OSAL_OBJTYPE_C` #define OSAL_OBJTYPE_C(

X) ((*osal_objtype_t*)(*X*))

Definition at line 174 of file common_types.h.

12.171.2.8 `OSAL_SIZE_C` #define OSAL_SIZE_C(

X) ((*size_t*)(*X*))

Definition at line 171 of file common_types.h.

12.171.2.9 `OSAL_STATUS_C` #define OSAL_STATUS_C(

X) ((*osal_status_t*)(*X*))

Definition at line 175 of file common_types.h.

12.171.3 Typedef Documentation

12.171.3.1 `cpuaddr` `typedef uintptr_t cpuaddr`

Definition at line 88 of file common_types.h.

12.171.3.2 `cpudiff` `typedef ptrdiff_t cpudiff`

Definition at line 90 of file common_types.h.

12.171.3.3 `cpusize` `typedef size_t cpusize`

Definition at line 89 of file common_types.h.

12.171.3.4 `int16` `typedef int16_t int16`

Definition at line 80 of file common_types.h.

12.171.3.5 `int32` `typedef int32_t int32`

Definition at line 81 of file common_types.h.

12.171.3.6 `int64` `typedef int64_t int64`

Definition at line 82 of file common_types.h.

12.171.3.7 `int8` `typedef int8_t int8`

Definition at line 79 of file common_types.h.

12.171.3.8 `intptr` `typedef intptr_t intptr`

Definition at line 87 of file common_types.h.

12.171.3.9 `OS_ArgCallback_t` `typedef void(* OS_ArgCallback_t) (osal_id_t object_id, void *arg)`

General purpose OSAL callback function.

This may be used by multiple APIS

Definition at line 143 of file common_types.h.

12.171.3.10 `osal_blockcount_t` `typedef size_t osal_blockcount_t`

A type used to represent a number of blocks or buffers

This is used with file system and queue implementations.

Definition at line 116 of file common_types.h.

12.171.3.11 `osal_id_t` `typedef uint32 osal_id_t`

A type to be used for OSAL resource identifiers. This typedef is backward compatible with the IDs from older versions of OSAL

Definition at line 108 of file common_types.h.

12.171.3.12 osal_index_t `typedef uint32 osal_index_t`

A type used to represent an index into a table structure

This is used when referring directly to a table index as opposed to an object ID. It is primarily intended for internal use, but is also output from public APIs such as [OS_ObjectIdToArrayIndex\(\)](#).

Definition at line 126 of file common_types.h.

12.171.3.13 osal_objtype_t `typedef uint32 osal_objtype_t`

A type used to represent the runtime type or category of an OSAL object

Definition at line 131 of file common_types.h.

12.171.3.14 osal_status_t `typedef int32 osal_status_t`

The preferred type to represent OSAL status codes defined in [osapi-error.h](#)

Definition at line 136 of file common_types.h.

12.171.3.15 uint16 `typedef uint16_t uint16`

Definition at line 84 of file common_types.h.

12.171.3.16 uint32 `typedef uint32_t uint32`

Definition at line 85 of file common_types.h.

12.171.3.17 uint64 `typedef uint64_t uint64`

Definition at line 86 of file common_types.h.

12.171.3.18 uint8 `typedef uint8_t uint8`

Definition at line 83 of file common_types.h.

12.171.4 Function Documentation

12.171.4.1 CompileTimeAssert() [1/9] `CompileTimeAssert (`

```
    sizeof(cpuaddr) >=sizeof(void *) ,  
    TypePtrWrongSize )
```

12.171.4.2 CompileTimeAssert() [2/9] `CompileTimeAssert (`

```
    sizeof(int16) = =2,  
    Typeint16WrongSize )
```

12.171.4.3 CompileTimeAssert() [3/9] `CompileTimeAssert (`

```
    sizeof(int32) = =4,  
    Typeint32WrongSize )
```

12.171.4.4 CompileTimeAssert() [4/9] `CompileTimeAssert (`
 `sizeof(int64) = 8,`
 `Typeint64WrongSize)`

12.171.4.5 CompileTimeAssert() [5/9] `CompileTimeAssert (`
 `sizeof(int8) = 1,`
 `Typeint8WrongSize)`

12.171.4.6 CompileTimeAssert() [6/9] `CompileTimeAssert (`
 `sizeof(uint16) = 2,`
 `TypeUint16WrongSize)`

12.171.4.7 CompileTimeAssert() [7/9] `CompileTimeAssert (`
 `sizeof(uint32) = 4,`
 `TypeUint32WrongSize)`

12.171.4.8 CompileTimeAssert() [8/9] `CompileTimeAssert (`
 `sizeof(uint64) = 8,`
 `TypeUint64WrongSize)`

12.171.4.9 CompileTimeAssert() [9/9] `CompileTimeAssert (`
 `sizeof(uint8) = 1,`
 `TypeUint8WrongSize)`

12.172 osal/src/os/inc/osapi-binsem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct [OS_bin_sem_prop_t](#)
OSAL binary semaphore properties.

Macros

- #define [OS_SEM_FULL](#) 1
Semaphore full state.
- #define [OS_SEM_EMPTY](#) 0
Semaphore empty state.

Functions

- int32 [OS_BinSemCreate](#) ([osal_id_t](#) *sem_id, const char *sem_name, [uint32](#) sem_initial_value, [uint32](#) options)
Creates a binary semaphore.
- int32 [OS_BinSemFlush](#) ([osal_id_t](#) sem_id)

Unblock all tasks pending on the specified semaphore.

- `int32 OS_BinSemGive (osal_id_t sem_id)`

Increment the semaphore value.

- `int32 OS_BinSemTake (osal_id_t sem_id)`

Decrement the semaphore value.

- `int32 OS_BinSemTimedWait (osal_id_t sem_id, uint32 msecs)`

Decrement the semaphore value with a timeout.

- `int32 OS_BinSemDelete (osal_id_t sem_id)`

Deletes the specified Binary Semaphore.

- `int32 OS_BinSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`

Find an existing semaphore ID by name.

- `int32 OS_BinSemGetInfo (osal_id_t sem_id, OS_bin_sem_prop_t *bin_prop)`

Fill a property object buffer with details regarding the resource.

12.172.1 Detailed Description

Declarations and prototypes for binary semaphores

12.173 osal/src/os/inc/osapi-bsp.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Functions

- `void OS_BSP_SetResourceTypeConfig (uint32 ResourceType, uint32 ConfigOptionValue)`
- `uint32 OS_BSP_GetResourceTypeConfig (uint32 ResourceType)`
- `uint32 OS_BSP_GetArgC (void)`
- `char *const * OS_BSP_GetArgV (void)`
- `void OS_BSP_SetExitCode (int32 code)`

12.173.1 Detailed Description

Declarations and prototypes for OSAL BSP

12.174 osal/src/os/inc/osapi-clock.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct `OS_time_t`

OSAL time interval structure.

Macros

- `#define OS_TIME_MAX ((OS_time_t){INT64_MAX})`
The maximum value for `OS_time_t`.
- `#define OS_TIME_ZERO ((OS_time_t){0})`
The zero value for `OS_time_t`.
- `#define OS_TIME_MIN ((OS_time_t){INT64_MIN})`
The minimum value for `OS_time_t`.

Enumerations

- `enum { OS_TIME_TICK_RESOLUTION_NS = 100, OS_TIME_TICKS_PER_SECOND = 1000000000 / OS_TIME_TICK_RESOLUTION_NS, OS_TIME_TICKS_PER_MSEC = 1000000 / OS_TIME_TICK_RESOLUTION_NS, OS_TIME_TICKS_PER_USEC = 1000 / OS_TIME_TICK_RESOLUTION_NS }`
Multippliers/divisors to convert ticks into standardized units.

Functions

- `int32 OS_GetLocalTime (OS_time_t *time_struct)`
Get the local time.
- `int32 OS_SetLocalTime (const OS_time_t *time_struct)`
Set the local time.
- `OS_time_t OS_TimeFromRelativeMilliseconds (int32 relative_msec)`
Gets an absolute time value relative to the current time.
- `int32 OS_TimeToRelativeMilliseconds (OS_time_t time)`
Gets a relative time value from an absolute time.
- `static int64 OS_TimeGetTotalSeconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to whole number of seconds.
- `static OS_time_t OS_TimeFromTotalSeconds (int64 tm)`
Get an `OS_time_t` interval object from an integer number of seconds.
- `static int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to millisecond units.
- `static OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of milliseconds.
- `static int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to microsecond units.
- `static OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of microseconds.
- `static int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to nanosecond units.
- `static OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`
Get an `OS_time_t` interval object from a integer number of nanoseconds.
- `static int64 OS_TimeGetFractionalPart (OS_time_t tm)`
Get subseconds portion (fractional part only) from an `OS_time_t` object.
- `static uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`
Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.
- `static uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`
Get milliseconds portion (fractional part only) from an `OS_time_t` object.
- `static uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`

- static `uint32 OS_TimeGetNanosecondsPart (OS_time_t tm)`

Get nanoseconds portion (fractional part only) from an `OS_time_t` object.
- static `OS_time_t OS_TimeAssembleFromNanoseconds (int64 seconds, uint32 nanoseconds)`

Assemble/Convert a number of seconds + nanoseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromMicroseconds (int64 seconds, uint32 microseconds)`

Assemble/Convert a number of seconds + microseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromMilliseconds (int64 seconds, uint32 milliseconds)`

Assemble/Convert a number of seconds + milliseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAssembleFromSubseconds (int64 seconds, uint32 subseconds)`

Assemble/Convert a number of seconds + subseconds into an `OS_time_t` interval.
- static `OS_time_t OS_TimeAdd (OS_time_t time1, OS_time_t time2)`

Computes the sum of two time intervals.
- static `OS_time_t OS_TimeSubtract (OS_time_t time1, OS_time_t time2)`

Computes the difference between two time intervals.
- static `bool OS_TimeEqual (OS_time_t time1, OS_time_t time2)`

Checks if two time values are equal.
- static `int8_t OS_TimeGetSign (OS_time_t time)`

Checks the sign of the time value.
- static `int8_t OS_TimeCompare (OS_time_t time1, OS_time_t time2)`

Compares two time values.

12.174.1 Detailed Description

Declarations and prototypes for osapi-clock module

12.174.2 Macro Definition Documentation

12.174.2.1 OS_TIME_MAX `#define OS_TIME_MAX ((OS_time_t) {INT64_MAX})`

The maximum value for `OS_time_t`.

This is the largest positive (future) time that is representable in an `OS_time_t` value.

Definition at line 56 of file osapi-clock.h.

12.174.2.2 OS_TIME_MIN `#define OS_TIME_MIN ((OS_time_t) {INT64_MIN})`

The minimum value for `OS_time_t`.

This is the largest negative (past) time that is representable in an `OS_time_t` value.

Definition at line 71 of file osapi-clock.h.

12.174.2.3 OS_TIME_ZERO `#define OS_TIME_ZERO ((OS_time_t) {0})`

The zero value for `OS_time_t`.

This is a reasonable initializer/placeholder value for an `OS_time_t`

Definition at line 63 of file osapi-clock.h.

12.174.3 Enumeration Type Documentation

12.174.3.1 anonymous enum anonymous enum

Multipliers/divisors to convert ticks into standardized units.

Various fixed conversion factor constants used by the conversion routines

A 100ns tick time allows max intervals of about +/- 14000 years in a 64-bit signed integer value.

Note

Applications should not directly use these values, but rather use conversion routines below to obtain standardized units (seconds/microseconds/etc).

Enumerator

OS_TIME_TICK_RESOLUTION_NS	
OS_TIME TICKS_PER_SECOND	
OS_TIME TICKS_PER_MSEC	
OS_TIME TICKS_PER_USEC	

Definition at line 84 of file osapi-clock.h.

12.175 osal/src/os/inc/osapi-common.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

TypeDefs

- **typedef int32(* OS_EventHandler_t)** (*OS_Event_t* event, *osal_id_t* object_id, void *data)
A callback routine for event handling.

Enumerations

- **enum OS_Event_t {**
 OS_EVENT_RESERVED = 0, *OS_EVENT_RESOURCE_ALLOCATED*, *OS_EVENT_RESOURCE_CREATED*,
 OS_EVENT_RESOURCE_DELETED,
 OS_EVENT_TASK_STARTUP, *OS_EVENT_MAX* **}**
A set of events that can be used with BSP event callback routines.

Functions

- **void OS_Application_Startup (void)**
Application startup.
- **void OS_Application_Run (void)**
Application run.
- **int32 OS_API_Init (void)**
Initialization of API.
- **void OS_API_Teardown (void)**
Teardown/de-initialization of OSAL 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.
- void `OS_ApplicationExit` (`int32` Status)
Exit/Abort the application.
- `int32 OS_RegisterEventHandler` (`OS_EventHandler_t` handler)
Callback routine registration.
- `size_t OS_strlen` (`const char *s`, `size_t maxlen`)
get string length

12.175.1 Detailed Description

Declarations and prototypes for general OSAL functions that are not part of a subsystem

12.175.2 Typedef Documentation

12.175.2.1 `OS_EventHandler_t` `typedef int32(* OS_EventHandler_t)(OS_Event_t event, osal_id_t object_id, void *data)`
A callback routine for event handling.

Parameters

<code>in</code>	<code>event</code>	The event that occurred
<code>in</code>	<code>object_id</code>	The associated object_id, or 0 if not associated with an object
<code>in, out</code>	<code>data</code>	An abstract data/context object associated with the event, or NULL.

Returns

`status` Execution status, see [OSAL Return Code Defines](#).

Definition at line 98 of file osapi-common.h.

12.175.3 Enumeration Type Documentation

12.175.3.1 `OS_Event_t` `enum OS_Event_t`

A set of events that can be used with BSP event callback routines.

Enumerator

<code>OS_EVENT_RESERVED</code>	no-op/reserved event id value
<code>OS_EVENT_RESOURCE_ALLOCATED</code>	resource/id has been newly allocated but not yet created. This event is invoked from WITHIN the locked region, in the context of the task which is allocating the resource. If the handler returns non-success, the error will be returned to the caller and the creation process is aborted.
<code>OS_EVENT_RESOURCE_CREATED</code>	resource/id has been fully created/finalized. Invoked outside locked region, in the context of the task which created the resource. Data object is not used, passed as NULL. Return value is ignored - this is for information purposes only.

Enumerator

OS_EVENT_RESOURCE_DELETED	resource/id has been deleted. Invoked outside locked region, in the context of the task which deleted the resource. Data object is not used, passed as NULL. Return value is ignored - this is for information purposes only.
OS_EVENT_TASK_STARTUP	New task is starting. Invoked outside locked region, in the context of the task which is currently starting, before the entry point is called. Data object is not used, passed as NULL. If the handler returns non-success, task startup is aborted and the entry point is not called.
OS_EVENT_MAX	placeholder for end of enum, not used

Definition at line 34 of file osapi-common.h.

12.176 osal/src/os/inc/osapi-condvar.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

Data Structures

- struct [OS_condvar_prop_t](#)
OSAL condition variable properties.

Functions

- [int32 OS_CondVarCreate \(osal_id_t *var_id, const char *var_name, uint32 options\)](#)
Creates a condition variable resource.
- [int32 OS_CondVarLock \(osal_id_t var_id\)](#)
Locks/Acquires the underlying mutex associated with a condition variable.
- [int32 OS_CondVarUnlock \(osal_id_t var_id\)](#)
Unlocks/Releases the underlying mutex associated with a condition variable.
- [int32 OS_CondVarSignal \(osal_id_t var_id\)](#)
Signals the condition variable resource referenced by var_id.
- [int32 OS_CondVarBroadcast \(osal_id_t var_id\)](#)
Broadcasts the condition variable resource referenced by var_id.
- [int32 OS_CondVarWait \(osal_id_t var_id\)](#)
Waits on the condition variable object referenced by var_id.
- [int32 OS_CondVarTimedWait \(osal_id_t var_id, const OS_time_t *abs_wakeup_time\)](#)
Time-limited wait on the condition variable object referenced by var_id.
- [int32 OS_CondVarDelete \(osal_id_t var_id\)](#)
Deletes the specified condition variable.
- [int32 OS_CondVarGetIdByName \(osal_id_t *var_id, const char *var_name\)](#)
Find an existing condition variable ID by name.
- [int32 OS_CondVarGetInfo \(osal_id_t var_id, OS_condvar_prop_t *condvar_prop\)](#)
Fill a property object buffer with details regarding the resource.

12.176.1 Detailed Description

Declarations and prototypes for condition variables

12.177 osal/src/os/inc/osapi-constants.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Macros

- `#define OS_PEND (-1)`
- `#define OS_CHECK (0)`
- `#define OS_OBJECT_ID_UNDEFINED ((osal_id_t) {0})`
Initializer for the osal_id_t type which will not match any valid value.
- `#define OS_OBJECT_CREATOR_ANY OS_OBJECT_ID_UNDEFINED`
Constant that may be passed to `OS_ForEachObject()`/`OS_ForEachObjectType()` to match any creator (i.e. get all objects)
- `#define OS_MAX_LOCAL_PATH_LEN (OS_MAX_PATH_LEN + OS_FS_PHYS_NAME_LEN)`
Maximum length of a local/native path name string.

12.177.1 Detailed Description

General constants for OSAL that are shared across subsystems

12.177.2 Macro Definition Documentation

12.177.2.1 OS_CHECK `#define OS_CHECK (0)`

Definition at line 35 of file osapi-constants.h.

12.177.2.2 OS_MAX_LOCAL_PATH_LEN `#define OS_MAX_LOCAL_PATH_LEN (OS_MAX_PATH_LEN + OS_FS_PHYS_NAME_LEN)`

Maximum length of a local/native path name string.

This is a concatenation of the OSAL virtual path with the system mount point or device name

Definition at line 54 of file osapi-constants.h.

12.177.2.3 OS_OBJECT_CREATOR_ANY `#define OS_OBJECT_CREATOR_ANY OS_OBJECT_ID_UNDEFINED`

Constant that may be passed to `OS_ForEachObject()`/`OS_ForEachObjectType()` to match any creator (i.e. get all objects)

Definition at line 46 of file osapi-constants.h.

12.177.2.4 OS_OBJECT_ID_UNDEFINED `#define OS_OBJECT_ID_UNDEFINED ((osal_id_t) {0})`

Initializer for the osal_id_t type which will not match any valid value.

Definition at line 40 of file osapi-constants.h.

12.177.2.5 OS_PEND `#define OS_PEND (-1)`

Definition at line 34 of file osapi-constants.h.

12.178 osal/src/os/inc/osapi-countsem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct [OS_count_sem_prop_t](#)

OSAL counting semaphore properties.

Functions

- [int32 OS_CountSemCreate \(osal_id_t *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options\)](#)
Creates a counting semaphore.
- [int32 OS_CountSemGive \(osal_id_t sem_id\)](#)
Increment the semaphore value.
- [int32 OS_CountSemTake \(osal_id_t sem_id\)](#)
Decrement the semaphore value.
- [int32 OS_CountSemTimedWait \(osal_id_t sem_id, uint32 msecs\)](#)
Decrement the semaphore value with timeout.
- [int32 OS_CountSemDelete \(osal_id_t sem_id\)](#)
Deletes the specified counting Semaphore.
- [int32 OS_CountSemGetIdByName \(osal_id_t *sem_id, const char *sem_name\)](#)
Find an existing semaphore ID by name.
- [int32 OS_CountSemGetInfo \(osal_id_t sem_id, OS_count_sem_prop_t *count_prop\)](#)
Fill a property object buffer with details regarding the resource.

12.178.1 Detailed Description

Declarations and prototypes for counting semaphores

12.179 osal/src/os/inc/osapi-dir.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct [os_dirent_t](#)

Directory entry.

Macros

- [#define OS_DIRENT_NAME\(x\) \(\(x\).FileName\)](#)

Access filename part of the dirent structure.

Functions

- `int32 OS_DirectoryOpen (osal_id_t *dir_id, const char *path)`
Opens a directory.
- `int32 OS_DirectoryClose (osal_id_t dir_id)`
Closes an open directory.
- `int32 OS_DirectoryRewind (osal_id_t dir_id)`
Rewinds an open directory.
- `int32 OS_DirectoryRead (osal_id_t 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.

12.179.1 Detailed Description

Declarations and prototypes for directories

12.179.2 Macro Definition Documentation

12.179.2.1 OS_DIRENTRY_NAME `#define OS_DIRENTRY_NAME (` `x) ((x).FileName)`

Access filename part of the dirent structure.

Definition at line 38 of file osapi-dir.h.

12.180 osal/src/os/inc/osapi-error.h File Reference

```
#include "common_types.h"
```

Macros

- `#define OS_ERROR_NAME_LENGTH 35`
Error string name length.
- `#define OS_STATUS_STRING_LENGTH 12`
Status converted to string length limit.
- `#define OS_SUCCESS (0)`
Successful execution.
- `#define OS_ERROR (-1)`
Failed execution.
- `#define OS_INVALID_POINTER (-2)`
Invalid pointer.
- `#define OS_ERROR_ADDRESS_MISALIGNED (-3)`
Address misalignment.
- `#define OS_ERROR_TIMEOUT (-4)`
Error timeout.
- `#define OS_INVALID_INT_NUM (-5)`
Invalid Interrupt number.

- #define **OS_SEM_FAILURE** (-6)
Semaphore failure.
- #define **OS_SEM_TIMEOUT** (-7)
Semaphore timeout.
- #define **OS_QUEUE_EMPTY** (-8)
Queue empty.
- #define **OS_QUEUE_FULL** (-9)
Queue full.
- #define **OS_QUEUE_TIMEOUT** (-10)
Queue timeout.
- #define **OS_QUEUE_INVALID_SIZE** (-11)
Queue invalid size.
- #define **OS_QUEUE_ID_ERROR** (-12)
Queue ID error.
- #define **OS_ERR_NAME_TOO_LONG** (-13)
*name length including null terminator greater than **OS_MAX_API_NAME***
- #define **OS_ERR_NO_FREE_IDS** (-14)
No free IDs.
- #define **OS_ERR_NAME_TAKEN** (-15)
Name taken.
- #define **OS_ERR_INVALID_ID** (-16)
Invalid ID.
- #define **OS_ERR_NAME_NOT_FOUND** (-17)
Name not found.
- #define **OS_ERR_SEM_NOT_FULL** (-18)
Semaphore not full.
- #define **OS_ERR_INVALID_PRIORITY** (-19)
Invalid priority.
- #define **OS_INVALID_SEM_VALUE** (-20)
Invalid semaphore value.
- #define **OS_ERR_FILE** (-27)
File error.
- #define **OS_ERR_NOT_IMPLEMENTED** (-28)
Not implemented.
- #define **OS_TIMER_ERR_INVALID_ARGS** (-29)
Timer invalid arguments.
- #define **OS_TIMER_ERR_TIMER_ID** (-30)
Timer ID error.
- #define **OS_TIMER_ERR_UNAVAILABLE** (-31)
Timer unavailable.
- #define **OS_TIMER_ERR_INTERNAL** (-32)
Timer internal error.
- #define **OS_ERR_OBJECT_IN_USE** (-33)
Object in use.
- #define **OS_ERR_BAD_ADDRESS** (-34)
Bad address.
- #define **OS_ERR_INCORRECT_OBJ_STATE** (-35)

- `#define OS_ERR_INCORRECT_OBJ_TYPE (-36)`
Incorrect object type.
- `#define OS_ERR_STREAM_DISCONNECTED (-37)`
Stream disconnected.
- `#define OS_ERR_OPERATION_NOT_SUPPORTED (-38)`
Requested operation not support on supplied object(s)
- `#define OS_ERR_INVALID_SIZE (-40)`
Invalid Size.
- `#define OS_ERR_OUTPUT_TOO_LARGE (-41)`
Size of output exceeds limit
- `#define OS_ERR_INVALID_ARGUMENT (-42)`
Invalid argument value (other than ID or size)
- `#define OS_FS_ERR_PATH_TOO_LONG (-103)`
FS path too long.
- `#define OS_FS_ERR_NAME_TOO_LONG (-104)`
FS name too long.
- `#define OS_FS_ERR_DRIVE_NOT_CREATED (-106)`
FS drive not created.
- `#define OS_FS_ERR_DEVICE_NOT_FREE (-107)`
FS device not free.
- `#define OS_FS_ERR_PATH_INVALID (-108)`
FS path invalid.

Typedefs

- `typedef char os_err_name_t[OS_ERROR_NAME_LENGTH]`
For the `OS_GetErrorName()` function, to ensure everyone is making an array of the same length.
- `typedef char os_status_string_t[OS_STATUS_STRING_LENGTH]`
For the `OS_StatusToString()` function, to ensure everyone is making an array of the same length.

Functions

- `static long OS_StatusToInteger (osal_status_t Status)`
Convert a status code to a native "long" type.
- `int32 OS_GetErrorName (int32 error_num, os_err_name_t *err_name)`
Convert an error number to a string.
- `char * OS_StatusToString (osal_status_t status, os_status_string_t *status_string)`
Convert status to a string.

12.180.1 Detailed Description

OSAL error code definitions

12.180.2 Macro Definition Documentation

12.180.2.1 OS_ERROR_NAME_LENGTH #define OS_ERROR_NAME_LENGTH 35

Error string name length.

The sizes of strings in OSAL functions are built with this limit in mind. Always check the uses of os_err_name_t when changing this value.

Definition at line 35 of file osapi-error.h.

12.180.2.2 OS_STATUS_STRING_LENGTH #define OS_STATUS_STRING_LENGTH 12

Status converted to string length limit.

Used for sizing os_status_string_t intended for use in printing osal_status_t values Sized to fit LONG_MIN including NULL termination

Definition at line 55 of file osapi-error.h.

12.180.3 Typedef Documentation**12.180.3.1 os_err_name_t** typedef char os_err_name_t[OS_ERROR_NAME_LENGTH]

For the [OS_GetErrorName\(\)](#) function, to ensure everyone is making an array of the same length.

Implementation note for developers:

The sizes of strings in OSAL functions are built with this [OS_ERROR_NAME_LENGTH](#) limit in mind. Always check the uses of os_err_name_t when changing this value.

Definition at line 47 of file osapi-error.h.

12.180.3.2 os_status_string_t typedef char os_status_string_t[OS_STATUS_STRING_LENGTH]

For the [OS_StatusToString\(\)](#) function, to ensure everyone is making an array of the same length.

Definition at line 61 of file osapi-error.h.

12.181 osal/src/os/inc/osapi-file.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

Data Structures

- struct [OS_file_prop_t](#)
OSAL file properties.
- struct [os_fstat_t](#)
File system status.

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_FILESTAT_MODE](#)(x) ((x). FileModeBits)

Access file stat mode bits.

- #define `OS_FILESTAT_ISDIR`(x) ((x). FileModeBits & `OS_FILESTAT_MODE_DIR`)
File stat is directory logical.
- #define `OS_FILESTAT_EXEC`(x) ((x). FileModeBits & `OS_FILESTAT_MODE_EXEC`)
File stat is executable logical.
- #define `OS_FILESTAT_WRITE`(x) ((x). FileModeBits & `OS_FILESTAT_MODE_WRITE`)
File stat is write enabled logical.
- #define `OS_FILESTAT_READ`(x) ((x). FileModeBits & `OS_FILESTAT_MODE_READ`)
File stat is read enabled logical.
- #define `OS_FILESTAT_SIZE`(x) ((x).FileSize)
Access file stat size field.
- #define `OS_FILESTAT_TIME`(x) (`OS_TimeGetTotalSeconds`((x).FileTime))
Access file stat time field as a whole number of seconds.

Enumerations

- enum { `OS_FILESTAT_MODE_EXEC` = 0x00001, `OS_FILESTAT_MODE_WRITE` = 0x00002, `OS_FILESTAT_MODE_READ` = 0x00004, `OS_FILESTAT_MODE_DIR` = 0x10000 }
File stat mode bits.
- enum `OS_file_flag_t`{ `OS_FILE_FLAG_NONE` = 0x00, `OS_FILE_FLAG_CREATE` = 0x01, `OS_FILE_FLAG_TRUNCATE` = 0x02 }
Flags that can be used with opening of a file (bitmask)

Functions

- int32 `OS_OpenCreate` (`osal_id_t` *filedes, const char *path, int32 flags, int32 access_mode)
Open or create a file.
- int32 `OS_close` (`osal_id_t` filedes)
Closes an open file handle.
- int32 `OS_read` (`osal_id_t` filedes, void *buffer, size_t nbytes)
Read from a file handle.
- int32 `OS_write` (`osal_id_t` filedes, const void *buffer, size_t nbytes)
Write to a file handle.
- int32 `OS_TimedReadAbs` (`osal_id_t` filedes, void *buffer, size_t nbytes, `OS_time_t` abstime)
File/Stream input read with a timeout.
- int32 `OS_TimedRead` (`osal_id_t` filedes, void *buffer, size_t nbytes, int32 timeout)
File/Stream input read with a timeout.
- int32 `OS_TimedWriteAbs` (`osal_id_t` filedes, const void *buffer, size_t nbytes, `OS_time_t` abstime)
File/Stream output write with a timeout.
- int32 `OS_TimedWrite` (`osal_id_t` filedes, const void *buffer, size_t nbytes, int32 timeout)
File/Stream output write with a timeout.
- int32 `OS_chmod` (const char *path, uint32 access_mode)
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` (`osal_id_t` 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 (osal_id_t 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.

12.181.1 Detailed Description

Declarations and prototypes for file objects

12.181.2 Macro Definition Documentation

12.181.2.1 OS_FILESTAT_EXEC `#define OS_FILESTAT_EXEC (`
 `x) ((x). FileModeBits & OS_FILESTAT_MODE_EXEC)`

File stat is executable logical.

Definition at line 92 of file osapi-file.h.

12.181.2.2 OS_FILESTAT_ISDIR `#define OS_FILESTAT_ISDIR (`
 `x) ((x). FileModeBits & OS_FILESTAT_MODE_DIR)`

File stat is directory logical.

Definition at line 90 of file osapi-file.h.

12.181.2.3 OS_FILESTAT_MODE `#define OS_FILESTAT_MODE (`
 `x) ((x). FileModeBits)`

Access file stat mode bits.

Definition at line 88 of file osapi-file.h.

12.181.2.4 OS_FILESTAT_READ `#define OS_FILESTAT_READ (`
 `x) ((x). FileModeBits & OS_FILESTAT_MODE_READ)`

File stat is read enabled logical.

Definition at line 96 of file osapi-file.h.

12.181.2.5 OS_FILESTAT_SIZE `#define OS_FILESTAT_SIZE (`
 `x) ((x).FileSize)`

Access file stat size field.

Definition at line 98 of file osapi-file.h.

```
12.181.2.6 OS_FILESTAT_TIME #define OS_FILESTAT_TIME (x) ((x).FileTime))
```

Access file stat time field as a whole number of seconds.

Definition at line 100 of file osapi-file.h.

```
12.181.2.7 OS_FILESTAT_WRITE #define OS_FILESTAT_WRITE (x) ((x). FileModeBits & OS_FILESTAT_MODE_WRITE)
```

File stat is write enabled logical.

Definition at line 94 of file osapi-file.h.

12.181.3 Enumeration Type Documentation

12.181.3.1 anonymous enum anonymous enum

File stat mode bits.

We must also define replacements for the stat structure's mode bits. This is currently just a small subset since the OSAL just presents a very simplified view of the filesystem to the upper layers. And since not all OS'es are POSIX, the more POSIX-specific bits are not relevant anyway.

Enumerator

OS_FILESTAT_MODE_EXEC	
OS_FILESTAT_MODE_WRITE	
OS_FILESTAT_MODE_READ	
OS_FILESTAT_MODE_DIR	

Definition at line 79 of file osapi-file.h.

12.181.3.2 OS_file_flag_t enum OS_file_flag_t

Flags that can be used with opening of a file (bitmask)

Enumerator

OS_FILE_FLAG_NONE	
OS_FILE_FLAG_CREATE	
OS_FILE_FLAG_TRUNCATE	

Definition at line 105 of file osapi-file.h.

12.182 osal/src/os/inc/osapi-filesystem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct [os_fsinfo_t](#)

OSAL file system info.

- struct [OS_statvfs_t](#)

Macros

- `#define OS_CHK_ONLY 0`
- `#define OS_REPAIR 1`

Functions

- `int32 OS_FileSysAddFixedMap (osal_id_t *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, size_t blocksize, osal_blockcount_t 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, size_t blocksize, osal_blockcount_t 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_FileSysStatVolume (const char *name, OS_statvfs_t *statbuf)`
Obtains information about size and free space 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 an OSAL Virtual file system path to a host Local path.
- `int32 OS_GetFsInfo (os_fsinfo_t *filesys_info)`
Returns information about the file system.

12.182.1 Detailed Description

Declarations and prototypes for file systems

12.182.2 Macro Definition Documentation

12.182.2.1 OS_CHK_ONLY `#define OS_CHK_ONLY 0`

Unused, API takes bool

Definition at line 31 of file osapi-filesystems.h.

12.182.2.2 OS_REPAIR `#define OS_REPAIR 1`

Unused, API takes bool

Definition at line 32 of file osapi-filesystems.h.

12.183 osal/src/os/inc/osapi-heap.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct `OS_heap_prop_t`

OSAL heap properties.

Functions

- `int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)`

Return current info on the heap.

12.183.1 Detailed Description

Declarations and prototypes for heap functions

12.184 osal/src/os/inc/osapi-idmap.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Macros

- `#define OS_OBJECT_INDEX_MASK 0xFFFF`
Object index mask.
- `#define OS_OBJECT_TYPE_SHIFT 16`
Object type shift.
- `#define OS_OBJECT_TYPE_UNDEFINED 0x00`
Object type undefined.
- `#define OS_OBJECT_TYPE_OS_TASK 0x01`
Object task type.
- `#define OS_OBJECT_TYPE_OS_QUEUE 0x02`
Object queue type.
- `#define OS_OBJECT_TYPE_OS_COUNTSEM 0x03`
Object counting semaphore type.
- `#define OS_OBJECT_TYPE_OS_BINSEM 0x04`
Object binary semaphore type.
- `#define OS_OBJECT_TYPE_OS_MUTEX 0x05`
Object mutex type.
- `#define OS_OBJECT_TYPE_OS_STREAM 0x06`
Object stream type.
- `#define OS_OBJECT_TYPE_OS_DIR 0x07`
Object directory type.
- `#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08`
Object timebase type.
- `#define OS_OBJECT_TYPE_OS_TIMECB 0x09`

Object timer callback type.

- #define OS_OBJECT_TYPE_OS_MODULE 0x0A
Object module type.
- #define OS_OBJECT_TYPE_OS_FILESYS 0x0B
Object file system type.
- #define OS_OBJECT_TYPE_OS_CONSOLE 0x0C
Object console type.
- #define OS_OBJECT_TYPE_OS_CONDVAR 0x0D
Object condition variable type.
- #define OS_OBJECT_TYPE_USER 0x10
Object user type.

Functions

- static unsigned long OS_ObjectIdToInteger (osal_id_t object_id)
Obtain an integer value corresponding to an object ID.
- static osal_id_t OS_ObjectIdFromInteger (unsigned long value)
Obtain an osal ID corresponding to an integer value.
- static bool OS_ObjectIdEqual (osal_id_t object_id1, osal_id_t object_id2)
Check two OSAL object ID values for equality.
- static bool OS_ObjectIdDefined (osal_id_t object_id)
Check if an object ID is defined.
- int32 OS_GetResourceName (osal_id_t object_id, char *buffer, size_t buffer_size)
Obtain the name of an object given an arbitrary object ID.
- osal_objtype_t OS_IdentifyObject (osal_id_t object_id)
Obtain the type of an object given an arbitrary object ID.
- int32 OS_ConvertToArrayIndex (osal_id_t object_id, osal_index_t *ArrayIndex)
Converts an abstract ID into a number suitable for use as an array index.
- int32 OS_ObjectIdToArrayIndex (osal_objtype_t idtype, osal_id_t object_id, osal_index_t *ArrayIndex)
Converts an abstract ID into a number suitable for use as an array index.
- void OS_ForEachObject (osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)
call the supplied callback function for all valid object IDs
- void OS_ForEachObjectOfType (osal_objtype_t objtype, osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)
call the supplied callback function for valid object IDs of a specific type

12.184.1 Detailed Description

Declarations and prototypes for object IDs

12.184.2 Macro Definition Documentation

12.184.2.1 OS_OBJECT_INDEX_MASK #define OS_OBJECT_INDEX_MASK 0xFFFF

Object index mask.

Definition at line 32 of file osapi-idmap.h.

12.184.2.2 OS_OBJECT_TYPE_SHIFT #define OS_OBJECT_TYPE_SHIFT 16
Object type shift.
Definition at line 33 of file osapi-idmap.h.

12.185 osal/src/os/inc/osapi-macros.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "osconfig.h"
#include "common_types.h"
#include "osapi-printf.h"
```

Macros

- #define **BUGREPORT**(...) OS_printf(__VA_ARGS__)
- #define **BUGCHECK**(cond, errcode)
 - Basic Bug-Checking macro.*
- #define **ARGCHECK**(cond, errcode)
 - Generic argument checking macro for non-critical values.*
- #define **LENGTHCHECK**(str, len, errcode) **ARGCHECK**(memchr(str, '\0', len), errcode)
 - String length limit check macro.*
- #define **BUGCHECK_VOID**(cond) **BUGCHECK**(cond,)
 - Bug-Check macro for void functions.*

12.185.1 Detailed Description

Macro definitions that are used across all OSAL subsystems

12.185.2 Macro Definition Documentation

12.185.2.1 ARGCHECK #define ARGCHECK(
 cond,
 errcode)

Value:

```
if (! (cond)) \
{ \
    return errcode; \
}
```

Generic argument checking macro for non-critical values.

This macro checks a conditional that is expected to be true, and return a value if it evaluates false.

ARGCHECK can be used to check for out of range or other invalid argument conditions which may (validly) occur at runtime and do not necessarily indicate bugs in the application.

These argument checks are NOT considered fatal errors. The application continues to run normally. This does not report the error on the console.

As such, ARGCHECK actions are always compiled in - not selectable at compile-time.

See also

[BUGCHECK](#) for checking critical values that indicate bugs

Definition at line 131 of file osapi-macros.h.

```
12.185.2.2 BUGCHECK #define BUGCHECK(  
    cond,  
    errcode )
```

Value:

```
if (! (cond))  
{  
    BUGREPORT("\n**BUG** %s():%d:check \'%s\' FAILED --> %s\n\n", __func__, __LINE__, #cond, #errcode); \  
    return errcode;  
}
```

Basic Bug-Checking macro.

This macro checks a conditional, and if it is FALSE, then it generates a report - which may in turn contain additional actions.

BUGCHECK should only be used for conditions which are critical and must always be true. If such a condition is ever false then it indicates a bug in the application which must be resolved. It may or may not be possible to continue operation if a bugcheck fails.

See also

[ARGCHECK](#) for checking non-critical values

Definition at line 105 of file osapi-macros.h.

```
12.185.2.3 BUGCHECK_VOID #define BUGCHECK_VOID(  
    cond ) BUGCHECK(cond, )
```

Bug-Check macro for void functions.

The basic BUGCHECK macro returns a value, which needs to be empty for functions that do not have a return value. In this case the second argument (errcode) is intentionally left blank.

Definition at line 155 of file osapi-macros.h.

```
12.185.2.4 BUGREPORT #define BUGREPORT(  
    ... ) OS_printf(__VA_ARGS__)
```

Definition at line 88 of file osapi-macros.h.

```
12.185.2.5 LENGTHCHECK #define LENGTHCHECK(  
    str,  
    len,  
    errcode ) ARGCHECK(memchr(str, '\0', len), errcode)
```

String length limit check macro.

This macro is a specialized version of ARGCHECK that confirms a string will fit into a buffer of the specified length, and return an error code if it will not.

Note

this uses ARGCHECK, thus treating a string too long as a normal runtime (i.e. non-bug) error condition with a typical error return to the caller.

Definition at line 146 of file osapi-macros.h.

12.186 osal/src/os/inc/osapi-module.h File Reference

```
#include "osconfig.h"  
#include "common_types.h"
```

Data Structures

- struct [OS_module_address_t](#)
OSAL module address properties.
- struct [OS_module_prop_t](#)
OSAL module properties.
- struct [OS_static_symbol_record_t](#)
Associates a single symbol name with a memory address.

Macros

- `#define OS_MODULE_FLAG_GLOBAL_SYMBOLS 0x00`
Requests [OS_ModuleLoad\(\)](#) to add the symbols to the global symbol table.
- `#define OS_MODULE_FLAG_LOCAL_SYMBOLS 0x01`
Requests [OS_ModuleLoad\(\)](#) to keep the symbols local/private to this module.

Functions

- `int32 OS_SymbolLookup (cpuaddr *symbol_address, const char *symbol_name)`
Find the Address of a Symbol.
- `int32 OS_ModuleSymbolLookup (osal_id_t module_id, cpuaddr *symbol_address, const char *symbol_name)`
Find the Address of a Symbol within a module.
- `int32 OS_SymbolTableDump (const char *filename, size_t size_limit)`
Dumps the system symbol table to a file.
- `int32 OS_ModuleLoad (osal_id_t *module_id, const char *module_name, const char *filename, uint32 flags)`
Loads an object file.
- `int32 OS_ModuleUnload (osal_id_t module_id)`
Unloads the module file.
- `int32 OS_ModuleInfo (osal_id_t module_id, OS_module_prop_t *module_info)`
Obtain information about a module.

12.186.1 Detailed Description

Declarations and prototypes for module subsystem

12.186.2 Macro Definition Documentation

12.186.2.1 OS_MODULE_FLAG_GLOBAL_SYMBOLS `#define OS_MODULE_FLAG_GLOBAL_SYMBOLS 0x00`

Requests [OS_ModuleLoad\(\)](#) to add the symbols to the global symbol table.

When supplied as the "flags" argument to [OS_ModuleLoad\(\)](#), this indicates that the symbols in the loaded module should be added to the global symbol table. This will make symbols in this library available for use when resolving symbols in future module loads.

This is the default mode of operation for [OS_ModuleLoad\(\)](#).

Note

On some operating systems, use of this option may make it difficult to unload the module in the future, if the symbols are in use by other entities.

Definition at line 49 of file osapi-module.h.

12.186.2.2 OS_MODULE_FLAG_LOCAL_SYMBOLS #define OS_MODULE_FLAG_LOCAL_SYMBOLS 0x01

Requests [OS_ModuleLoad\(\)](#) to keep the symbols local/private to this module.

When supplied as the "flags" argument to [OS_ModuleLoad\(\)](#), this indicates that the symbols in the loaded module should NOT be added to the global symbol table. This means the symbols in the loaded library will not be available for use by other modules.

Use this option is recommended for cases where no other entities will need to reference symbols within this module. This helps ensure that the module can be more safely unloaded in the future, by preventing other modules from binding to it. It also helps reduce the likelihood of symbol name conflicts among modules.

Note

To look up symbols within a module loaded with this flag, use [OS_SymbolLookupInModule\(\)](#) instead of [OS_SymbolLookup\(\)](#). Also note that references obtained using this method are not tracked by the OS; the application must ensure that all references obtained in this manner have been cleaned up/released before unloading the module.

Definition at line 71 of file osapi-module.h.

12.187 osal/src/os/inc/osapi-mutex.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct [OS_mut_sem_prop_t](#)

OSAL mutex properties.

Functions

- int32 [OS_MutSemCreate](#) (osal_id_t *sem_id, const char *sem_name, uint32 options)
Creates a mutex semaphore.
- int32 [OS_MutSemGive](#) (osal_id_t sem_id)
Releases the mutex object referenced by sem_id.
- int32 [OS_MutSemTake](#) (osal_id_t sem_id)
Acquire the mutex object referenced by sem_id.
- int32 [OS_MutSemDelete](#) (osal_id_t sem_id)
Deletes the specified Mutex Semaphore.
- int32 [OS_MutSemGetIdByName](#) (osal_id_t *sem_id, const char *sem_name)
Find an existing mutex ID by name.
- int32 [OS_MutSemGetInfo](#) (osal_id_t sem_id, [OS_mut_sem_prop_t](#) *mut_prop)
Fill a property object buffer with details regarding the resource.

12.187.1 Detailed Description

Declarations and prototypes for mutexes

12.188 osal/src/os/inc/osapi-network.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Functions

- `int32 OS_NetworkGetID (void)`
Gets the network ID of the local machine.
- `int32 OS_NetworkGetHostName (char *host_name, size_t name_len)`
Gets the local machine network host name.

12.188.1 Detailed Description

Declarations and prototypes for network subsystem

12.189 osal/src/os/inc/osapi-printf.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Functions

- `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 from OS_printf.
- `void OS_printf_enable (void)`
This function enables the output from OS_printf.

12.189.1 Detailed Description

Declarations and prototypes for printf/console output

12.190 osal/src/os/inc/osapi-queue.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- `struct OS_queue_prop_t`
OSAL queue properties.

Functions

- `int32 OS_QueueCreate (osal_id_t *queue_id, const char *queue_name, osal_blockcount_t queue_depth, size_t data_size, uint32 flags)`
Create a message queue.
- `int32 OS_QueueDelete (osal_id_t queue_id)`
Deletes the specified message queue.
- `int32 OS_QueueGet (osal_id_t queue_id, void *data, size_t size, size_t *size_copied, int32 timeout)`
Receive a message on a message queue.
- `int32 OS_QueuePut (osal_id_t queue_id, const void *data, size_t size, uint32 flags)`
Put a message on a message queue.

- int32 OS_QueueGetIdByName (`osal_id_t` *queue_id, const char *queue_name)
Find an existing queue ID by name.
- int32 OS_QueueGetInfo (`osal_id_t` queue_id, `OS_queue_prop_t` *queue_prop)
Fill a property object buffer with details regarding the resource.

12.190.1 Detailed Description

Declarations and prototypes for queue subsystem

12.191 osal/src/os/inc/osapi-select.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

Data Structures

- struct `OS_FdSet`
An abstract structure capable of holding several OSAL IDs.

Enumerations

- enum `OS_StreamState_t` {
 `OS_STREAM_STATE_BOUND` = 0x01, `OS_STREAM_STATE_CONNECTED` = 0x02, `OS_STREAM_STATE_READABLE` = 0x04, `OS_STREAM_STATE_WRITABLE` = 0x08,
 `OS_STREAM_STATE_LISTENING` = 0x10 }

For the `OS_SelectSingle()` function's in/out `StateFlags` parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.

Functions

- int32 `OS_SelectMultipleAbs` (`OS_FdSet` *ReadSet, `OS_FdSet` *WriteSet, `OS_time_t` abs_timeout)
Wait for events across multiple file handles.
- int32 `OS_SelectMultiple` (`OS_FdSet` *ReadSet, `OS_FdSet` *WriteSet, int32 msecs)
Wait for events across multiple file handles.
- int32 `OS_SelectSingleAbs` (`osal_id_t` objid, uint32 *StateFlags, `OS_time_t` abs_timeout)
Wait for events on a single file handle.
- int32 `OS_SelectSingle` (`osal_id_t` 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, `osal_id_t` objid)
Add an ID to an FdSet structure.
- int32 `OS_SelectFdClear` (`OS_FdSet` *Set, `osal_id_t` objid)
Clear an ID from an FdSet structure.
- bool `OS_SelectFdsSet` (const `OS_FdSet` *Set, `osal_id_t` objid)
Check if an FdSet structure contains a given ID.

12.191.1 Detailed Description

Declarations and prototypes for select abstraction

12.191.2 Enumeration Type Documentation

12.191.2.1 OS_StreamState_t enum OS_StreamState_t

For the [OS_SelectSingle\(\)](#) function's in/out StateFlags parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.

See also

[OS_SelectSingle\(\)](#)

Enumerator

OS_STREAM_STATE_BOUND	whether the stream is bound
OS_STREAM_STATE_CONNECTED	whether the stream is connected
OS_STREAM_STATE_READABLE	whether the stream is readable
OS_STREAM_STATE_WRITABLE	whether the stream is writable
OS_STREAM_STATE_LISTENING	whether the stream is listening

Definition at line 56 of file osapi-select.h.

12.192 osal/src/os/inc/osapi-shell.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Functions

- int32 [OS_ShellOutputToFile](#) (const char *Cmd, [osal_id_t](#) filedes)

Executes the command and sends output to a file.

12.192.1 Detailed Description

Declarations and prototypes for shell abstraction

12.193 osal/src/os/inc/osapi-sockets.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

Data Structures

- union [OS_SockAddrData_t](#)

Storage buffer for generic network address.
- struct [OS_SockAddr_t](#)

Encapsulates a generic network address.

- struct `OS_socket_prop_t`
Encapsulates socket properties.

Macros

- `#define OS SOCKADDR_MAX_LEN 28`

Enumerations

- enum `OS_SocketDomain_t`{ `OS_SocketDomain_INVALID`, `OS_SocketDomain_INET`, `OS_SocketDomain_INET6`,
`OS_SocketDomain_MAX` }
Socket domain.
- enum `OS_SocketType_t`{ `OS_SocketType_INVALID`, `OS_SocketType_DATAGRAM`, `OS_SocketType_STREAM`,
`OS_SocketType_MAX` }
Socket type.
- enum `OS_SocketShutdownMode_t`{ `OS_SocketShutdownMode_NONE` = 0, `OS_SocketShutdownMode_SHUT_READ` = 1, `OS_SocketShutdownMode_SHUT_WRITE` = 2, `OS_SocketShutdownMode_SHUT_RDWR` = 3 }
Shutdown Mode.

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, size_t 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 (osal_id_t *sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)`
Opens a socket.
- `int32 OS_SocketBind (osal_id_t sock_id, const OS_SockAddr_t *Addr)`
Binds a socket to a given local address and enter listening (server) mode.
- `int32 OS_SocketListen (osal_id_t sock_id)`
Places the specified socket into a listening state.
- `int32 OS_SocketBindAddress (osal_id_t sock_id, const OS_SockAddr_t *Addr)`
Binds a socket to a given local address.
- `int32 OS_SocketConnectAbs (osal_id_t sock_id, const OS_SockAddr_t *Addr, OS_time_t abs_timeout)`
Connects a socket to a given remote address.
- `int32 OS_SocketConnect (osal_id_t sock_id, const OS_SockAddr_t *Addr, int32 timeout)`
Connects a socket to a given remote address.
- `int32 OS_SocketShutdown (osal_id_t sock_id, OS_SocketShutdownMode_t Mode)`
Implement graceful shutdown of a stream socket.
- `int32 OS_SocketAcceptAbs (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, OS_time_t abs_timeout)`
Waits for and accept the next incoming connection on the given socket.
- `int32 OS_SocketAccept (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, int32 timeout)`

Waits for and accept the next incoming connection on the given socket.

- `int32 OS_SocketRecvFromAbs (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, OS_time_t abs_timeout)`

Reads data from a message-oriented (datagram) socket.

- `int32 OS_SocketRecvFrom (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, int32 timeout)`

Reads data from a message-oriented (datagram) socket.

- `int32 OS_SocketSendTo (osal_id_t sock_id, const void *buffer, size_t buflen, const OS_SockAddr_t *RemoteAddr)`

Sends data to a message-oriented (datagram) socket.

- `int32 OS_SocketGetIdByName (osal_id_t *sock_id, const char *sock_name)`

Gets an OSAL ID from a given name.

- `int32 OS_SocketGetInfo (osal_id_t sock_id, OS_socket_prop_t *sock_prop)`

Gets information about an OSAL Socket ID.

12.193.1 Detailed Description

Declarations and prototypes for sockets abstraction

12.193.2 Macro Definition Documentation

12.193.2.1 OS SOCKADDR_MAX_LEN #define OS SOCKADDR_MAX_LEN 28

Definition at line 46 of file osapi-sockets.h.

12.193.3 Enumeration Type Documentation

12.193.3.1 OS_SocketDomain_t enum OS_SocketDomain_t

Socket domain.

Enumerator

<code>OS_SocketDomain_INVALID</code>	Invalid.
<code>OS_SocketDomain_INET</code>	IPv4 address family, most commonly used)
<code>OS_SocketDomain_INET6</code>	IPv6 address family, depends on OS/network stack support.
<code>OS_SocketDomain_MAX</code>	Maximum.

Definition at line 61 of file osapi-sockets.h.

12.193.3.2 OS_SocketShutdownMode_t enum OS_SocketShutdownMode_t

Shutdown Mode.

Enumerator

<code>OS_SocketShutdownMode_NONE</code>	Reserved value, no effect.
<code>OS_SocketShutdownMode_SHUT_READ</code>	Disable future reading.
<code>OS_SocketShutdownMode_SHUT_WRITE</code>	Disable future writing.
<code>OS_SocketShutdownMode_SHUT_RDWR</code>	Disable future reading or writing.

Definition at line 80 of file osapi-sockets.h.

12.193.3.3 OS_SocketType_t enum OS_SocketType_t

Socket type.

Enumerator

OS_SocketType_INVALID	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	Maximum.

Definition at line 70 of file osapi-sockets.h.

12.194 osal/src/os/inc/osapi-task.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct OS_task_prop_t

OSAL task properties.

Macros

- #define OS_MAX_TASK_PRIORITY 255
Upper limit for OSAL task priorities.
- #define OS_FP_ENABLED 1
Floating point enabled state for a task.
- #define OSAL_PRIORITY_C(X) ((osal_priority_t) {X})
- #define OSAL_STACKPTR_C(X) ((osal_stackptr_t) {X})
- #define OSAL_TASK_STACK_ALLOCATE OSAL_STACKPTR_C(NULL)

Typedefs

- typedef uint8_t osal_priority_t
Type to be used for OSAL task priorities.
- typedef void * osal_stackptr_t
Type to be used for OSAL stack pointer.
- typedef void osal_task
For task entry point.

Functions

- typedef osal_task ((*osal_task_entry))(void)
For task entry point.
- int32 OS_TaskCreate (osal_id_t *task_id, const char *task_name, osal_task_entry function_pointer, osal_stackptr_t stack_pointer, size_t stack_size, osal_priority_t priority, uint32 flags)

- Creates a task and starts running it.*
- `int32 OS_TaskDelete (osal_id_t 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 (osal_id_t task_id, osal_priority_t new_priority)`
Sets the given task to a new priority.
 - `osal_id_t OS_TaskGetId (void)`
Obtain the task id of the calling task.
 - `int32 OS_TaskGetIdByName (osal_id_t *task_id, const char *task_name)`
Find an existing task ID by name.
 - `int32 OS_TaskGetInfo (osal_id_t task_id, OS_task_prop_t *task_prop)`
Fill a property object buffer with details regarding the resource.
 - `int32 OS_TaskFindIdBySystemData (osal_id_t *task_id, const void *sysdata, size_t sysdata_size)`
Reverse-lookup the OSAL task ID from an operating system ID.

12.194.1 Detailed Description

Declarations and prototypes for task abstraction

12.194.2 Macro Definition Documentation

12.194.2.1 OS_FP_ENABLED `#define OS_FP_ENABLED 1`

Floating point enabled state for a task.

Definition at line 35 of file osapi-task.h.

12.194.2.2 OS_MAX_TASK_PRIORITY `#define OS_MAX_TASK_PRIORITY 255`

Upper limit for OSAL task priorities.

Definition at line 32 of file osapi-task.h.

12.194.2.3 OSAL_PRIORITY_C `#define OSAL_PRIORITY_C(X) ((osal_priority_t) {X})`

Definition at line 46 of file osapi-task.h.

12.194.2.4 OSAL_STACKPTR_C `#define OSAL_STACKPTR_C(X) ((osal_stackptr_t) {X})`

Definition at line 53 of file osapi-task.h.

12.194.2.5 OSAL_TASK_STACK_ALLOCATE `#define OSAL_TASK_STACK_ALLOCATE OSAL_STACKPTR_C(NULL)`

Definition at line 54 of file osapi-task.h.

12.194.3 Typedef Documentation

12.194.3.1 `osal_priority_t` typedef uint8_t osal_priority_t

Type to be used for OSAL task priorities.

OSAL priorities are in reverse order, and range from 0 (highest; will preempt all other tasks) to 255 (lowest; will not preempt any other task).

Definition at line 44 of file osapi-task.h.

12.194.3.2 `osal_stackptr_t` typedef void* osal_stackptr_t

Type to be used for OSAL stack pointer.

Definition at line 51 of file osapi-task.h.

12.194.3.3 `osal_task` typedef void osal_task

For task entry point.

Definition at line 68 of file osapi-task.h.

12.194.4 Function Documentation

12.194.4.1 `osal_task()` typedef osal_task (`(*) (void) osal_task_entry)`

For task entry point.

12.195 osal/src/os/inc/osapi-timebase.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct `OS_timebase_prop_t`

Time base properties.

TypeDefs

- `typedef uint32(* OS_TimerSync_t) (osal_id_t timer_id)`

Timer sync.

Functions

- `int32 OS_TimeBaseCreate (osal_id_t *timebase_id, const char *timebase_name, OS_TimerSync_t external_sync)`
Create an abstract Time Base resource.
- `int32 OS_TimeBaseSet (osal_id_t timebase_id, uint32 start_time, uint32 interval_time)`
Sets the tick period for simulated time base objects.
- `int32 OS_TimeBaseDelete (osal_id_t timebase_id)`
Deletes a time base object.

- `int32 OS_TimeBaseGetIdByName (osal_id_t *timebase_id, const char *timebase_name)`
Find the ID of an existing time base resource.
- `int32 OS_TimeBaseGetInfo (osal_id_t timebase_id, OS_timebase_prop_t *timebase_prop)`
Obtain information about a timebase resource.
- `int32 OS_TimeBaseGetFreeRun (osal_id_t timebase_id, uint32 *freerun_val)`
Read the value of the timebase free run counter.

12.195.1 Detailed Description

Declarations and prototypes for timebase abstraction

12.195.2 Typedef Documentation

12.195.2.1 OS_TimerSync_t `typedef uint32(* OS_TimerSync_t) (osal_id_t timer_id)`

Timer sync.

Definition at line 34 of file osapi-timebase.h.

12.196 osal/src/os/inc/osapi-timer.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct `OS_timer_prop_t`
Timer properties.

TypeDefs

- `typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)`
Timer callback.

Functions

- `int32 OS_TimerCreate (osal_id_t *timer_id, const char *timer_name, uint32 *clock_accuracy, OS_TimerCallback_t callback_ptr)`
Create a timer object.
- `int32 OS_TimerAdd (osal_id_t *timer_id, const char *timer_name, osal_id_t timebase_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`
Add a timer object based on an existing TimeBase resource.
- `int32 OS_TimerSet (osal_id_t timer_id, uint32 start_time, uint32 interval_time)`
Configures a periodic or one shot timer.
- `int32 OS_TimerDelete (osal_id_t timer_id)`
Deletes a timer resource.
- `int32 OS_TimerGetIdByName (osal_id_t *timer_id, const char *timer_name)`
Locate an existing timer resource by name.
- `int32 OS_TimerGetInfo (osal_id_t timer_id, OS_timer_prop_t *timer_prop)`
Gets information about an existing timer.

12.196.1 Detailed Description

Declarations and prototypes for timer abstraction (app callbacks)

12.196.2 Typedef Documentation

12.196.2.1 OS_TimerCallback_t `typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)`

Timer callback.

Definition at line 34 of file osapi-timer.h.

12.197 osal/src/os/inc/osapi-version.h File Reference

```
#include "common_types.h"
```

Macros

- `#define OS_BUILD_NUMBER 97`
: Development: Release name for current development cycle.
- `#define OS_BUILD_BASELINE "equuleus-rc1"`
: Development: Code name for the current build
- `#define OS_BUILD_DEV_CYCLE "equuleus-rc2"`
: Development: Release name for current development cycle.
- `#define OS_BUILD_CODENAME "Equuleus"`
: Development: Code name for the current build
- `#define OS_MAJOR_VERSION 5`
Major version number.
- `#define OS_MINOR_VERSION 0`
Minor version number.
- `#define OS_REVISION 0`
Revision version number. Value of 99 indicates a development version.
- `#define OS_LAST_OFFICIAL "v5.0.0"`
Last official release.
- `#define OS_MISSION_REV 0xFF`
Mission revision.
- `#define OS_STR_HELPER(x) #x`
Helper function to concatenate strings from integer.
- `#define OS_STR(x) OS_STR_HELPER(x)`
Helper function to concatenate strings from integer.
- `#define OS_VERSION OS_BUILD_BASELINE "+dev" OS_STR(OS_BUILD_NUMBER)`
Development Build Version Number.
- `#define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION * 100) + OS_REVISION)`
Combines the revision components into a single value.
- `#define OS_CFG_MAX_VERSION_STR_LEN 256`
Max Version String length.

Functions

- const char * [OS_GetVersionString](#) (void)
- const char * [OS_GetVersionCodeName](#) (void)
- void [OS_GetVersionNumber](#) ([uint8](#) VersionNumbers[4])
Obtain the OSAL numeric version number.
- [uint32 OS_GetBuildNumber](#) (void)
Obtain the OSAL library numeric build number.

12.197.1 Detailed Description

Provide version identifiers for Operating System Abstraction Layer

Note

OSAL follows the same version semantics as cFS, which in turn is based on the Semantic Versioning 2.0 Specification. For more information, see the documentation provided with cFE.

12.197.2 Macro Definition Documentation

12.197.2.1 OS_BUILD_BASELINE [#define OS_BUILD_BASELINE "equuleus-rc1"](#)

Definition at line 38 of file osapi-version.h.

12.197.2.2 OS_BUILD_CODENAME [#define OS_BUILD_CODENAME "Equuleus"](#)

: Development: Code name for the current build

Definition at line 40 of file osapi-version.h.

12.197.2.3 OS_BUILD_DEV_CYCLE [#define OS_BUILD_DEV_CYCLE "equuleus-rc2"](#)

: Development: Release name for current development cycle.

Definition at line 39 of file osapi-version.h.

12.197.2.4 OS_BUILD_NUMBER [#define OS_BUILD_NUMBER 97](#)

Definition at line 37 of file osapi-version.h.

12.197.2.5 OS_CFG_MAX_VERSION_STR_LEN [#define OS_CFG_MAX_VERSION_STR_LEN 256](#)

Max Version String length.

Maximum length that an OSAL version string can be.

Definition at line 154 of file osapi-version.h.

12.197.2.6 OS_LAST_OFFICIAL [#define OS_LAST_OFFICIAL "v5.0.0"](#)

Last official release.

Definition at line 52 of file osapi-version.h.

12.197.2.7 OS_MAJOR_VERSION #define OS_MAJOR_VERSION 5
Major version number.
Definition at line 45 of file osapi-version.h.

12.197.2.8 OS_MINOR_VERSION #define OS_MINOR_VERSION 0
Minor version number.
Definition at line 46 of file osapi-version.h.

12.197.2.9 OS_MISSION_REV #define OS_MISSION_REV 0xFF
Mission revision.
Reserved for mission use to denote patches/customizations as needed. Values 1-254 are reserved for mission use to denote patches/customizations as needed. NOTE: Reserving 0 and 0xFF for cFS open-source development use (pending resolution of nasa/cFS#440)
Definition at line 61 of file osapi-version.h.

12.197.2.10 OS_REVISION #define OS_REVISION 0
Revision version number. Value of 99 indicates a development version.
Definition at line 47 of file osapi-version.h.

12.197.2.11 OS_STR #define OS_STR(
 x) OS_STR_HELPER(x)
Helper function to concatenate strings from integer.
Definition at line 67 of file osapi-version.h.

12.197.2.12 OS_STR_HELPER #define OS_STR_HELPER(
 x) #x
Helper function to concatenate strings from integer.
Definition at line 66 of file osapi-version.h.

12.197.2.13 OS_VERSION #define OS_VERSION OS_BUILD_BASELINE "+dev" OS_STR(OS_BUILD_NUMBER)
Development Build Version Number.
Baseline git tag + Number of commits since baseline.
Definition at line 72 of file osapi-version.h.

12.197.2.14 OSAL_API_VERSION #define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION
* 100) + OS_REVISION)
Combines the revision components into a single value.
Applications can check against this number
e.g. "#if OSAL_API_VERSION >= 40100" would check if some feature added in OSAL 4.1 is present.
Definition at line 79 of file osapi-version.h.

12.197.3 Function Documentation

```
12.197.3.1 OS_GetBuildNumber() uint32 OS_GetBuildNumber (
    void )
```

Obtain the OSAL library numeric build number.

The build number is a monotonically increasing number that (coarsely) reflects the number of commits/changes that have been merged since the epoch release. During development cycles this number should increase after each subsequent merge/modification.

Like other version information, this is a fixed number assigned at compile time.

Returns

The OSAL library build number

```
12.197.3.2 OS_GetVersionCodeName() const char* OS_GetVersionCodeName (
    void )
```

Gets the OSAL version code name

All NASA CFE/CFS components (including CFE framework, OSAL and PSP) that work together will share the same code name.

Returns

OSAL code name. This is a fixed value string and is never NULL.

```
12.197.3.3 OS_GetVersionNumber() void OS_GetVersionNumber (
    uint8 VersionNumbers[4] )
```

Obtain the OSAL numeric version number.

This retrieves the numeric OSAL version identifier as an array of 4 uint8 values.

The array of numeric values is in order of precedence: [0] = Major Number [1] = Minor Number [2] = Revision Number [3] = Mission Revision

The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. 0 indicates an official release 1-254 local patch level (reserved for mission use) 255 indicates a development build

Parameters

<code>out</code>	<code>VersionNumbers</code>	A fixed-size array to be filled with the version numbers
------------------	-----------------------------	--

```
12.197.3.4 OS_GetVersionString() const char* OS_GetVersionString (
    void )
```

Gets the OSAL version/baseline ID as a string

This returns the content of the `OS_VERSION` macro defined above, and is specifically just the baseline and development build ID (if applicable), without any extra info.

Returns

Basic version identifier. This is a fixed value string and is never NULL.

12.198 osal/src/os/inc/osapi.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
```

```
#include "common_types.h"
#include "osapi-version.h"
#include "osconfig.h"
#include "osapi-binsem.h"
#include "osapi-clock.h"
#include "osapi-common.h"
#include "osapi-condvar.h"
#include "osapi-constants.h"
#include "osapi-countsem.h"
#include "osapi-dir.h"
#include "osapi-error.h"
#include "osapi-file.h"
#include "osapi-fs.h"
#include "osapi-heap.h"
#include "osapi-macros.h"
#include "osapi-idmap.h"
#include "osapi-module.h"
#include "osapi-mutex.h"
#include "osapi-network.h"
#include "osapi-printf.h"
#include "osapi-queue.h"
#include "osapi-select.h"
#include "osapi-shell.h"
#include "osapi-sockets.h"
#include "osapi-task.h"
#include "osapi-timebase.h"
#include "osapi-timer.h"
#include "osapi-bsp.h"
```

12.198.1 Detailed Description

Purpose: Contains functions prototype definitions and variables declarations for the OS Abstraction Layer, Core OS module

12.199 psp/fsw/inc/cfe_psp.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_cache_api.h"
#include "cfe_psp_cds_api.h"
#include "cfe_psp_eepromaccess_api.h"
#include "cfe_psp_error.h"
#include "cfe_psp_exception_api.h"
#include "cfe_psp_id_api.h"
#include "cfe_psp_memaccess_api.h"
#include "cfe_psp_memrange_api.h"
#include "cfe_psp_port_api.h"
#include "cfe_psp_ssr_api.h"
#include "cfe_psp_timetick_api.h"
#include "cfe_psp_version_api.h"
#include "cfe_psp_watchdog_api.h"
```

Functions

- void **CFE_PSP_Main** (void)
PSP Entry Point to initialize the OSAL and start up the cFE.

12.199.1 Function Documentation

12.199.1.1 CFE_PSP_Main() void CFE_PSP_Main (
 void)

PSP Entry Point to initialize the OSAL and start up the cFE.

This is the entry point that the real-time OS calls to start our software. This routine will do any BSP/OS-specific setup, then call the entry point of the flight software (i.e. the cFE main entry point).

Note

The flight software (i.e. cFE) should not call this routine.

12.200 psp/fsw/inc/cfe_psp_cache_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- void **CFE_PSP_FlushCaches** (uint32 type, void *address, uint32 size)
This is a BSP-specific cache flush routine.

12.200.1 Function Documentation

12.200.1.1 CFE_PSP_FlushCaches() void CFE_PSP_FlushCaches (
 uint32 type,
 void * address,
 uint32 size)

This is a BSP-specific cache flush routine.

Provides a common interface to flush the processor caches. This routine is in the BSP because it is sometimes implemented in hardware and sometimes taken care of by the RTOS.

Parameters

in	<i>type</i>	
in	<i>address</i>	
in	<i>size</i>	

12.201 psp/fsw/inc/cfe_psp_cds_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
```

```
#include "cfe_psp_error.h"
```

Functions

- `int32 CFE_PSP_GetCDSSize (uint32 *SizeOfCDS)`
Fetches the size of the OS Critical Data Store area.
- `int32 CFE_PSP_WriteToCDS (const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)`
Writes to the CDS Block.
- `int32 CFE_PSP_ReadFromCDS (void *PtrToDataFromRead, uint32 CDSOffset, uint32 NumBytes)`
Reads from the CDS Block.

12.201.1 Function Documentation

12.201.1.1 CFE_PSP_GetCDSSize() `int32 CFE_PSP_GetCDSSize (` `uint32 * SizeOfCDS)`

Fetches the size of the OS Critical Data Store area.

Parameters

<code>out</code>	<code>SizeOfCDS</code>	Pointer to the variable that will store the size of the CDS
------------------	------------------------	---

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.201.1.2 CFE_PSP_ReadFromCDS() `int32 CFE_PSP_ReadFromCDS (` `void * PtrToDataFromRead,` `uint32 CDSOffset,` `uint32 NumBytes)`

Reads from the CDS Block.

Parameters

<code>out</code>	<code>PtrToDataFromRead</code>	Pointer to the location that will store the data to be read from the CDS
<code>in</code>	<code>CDSOffset</code>	CDS offset
<code>in</code>	<code>NumBytes</code>	Number of bytes to read

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.201.1.3 CFE_PSP_WriteToCDS() `int32 CFE_PSP_WriteToCDS (` `const void * PtrToDataToWrite,` `uint32 CDSOffset,` `uint32 NumBytes)`

Writes to the CDS Block.

Parameters

in	<i>PtrToDataToWrite</i>	Pointer to the data that will be written to the CDS
in	<i>CDSOffset</i>	CDS offset
in	<i>NumBytes</i>	Number of bytes to write

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.202 psp/fsw/inc/cfe_psp_eepromaccess_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- **int32 CFE_PSP_EepromWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)**
Write one byte (ByteValue) to EEPROM address MemoryAddress.
- **int32 CFE_PSP_EepromWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)**
Write two bytes (uint16Value) to EEPROM address MemoryAddress.
- **int32 CFE_PSP_EepromWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)**
Write four bytes (uint32Value) to EEPROM address MemoryAddress.
- **int32 CFE_PSP_EepromWriteEnable (uint32 Bank)**
Enable the EEPROM for write operation.
- **int32 CFE_PSP_EepromWriteDisable (uint32 Bank)**
Disable the EEPROM from write operation.
- **int32 CFE_PSP_EepromPowerUp (uint32 Bank)**
Power up the EEPROM.
- **int32 CFE_PSP_EepromPowerDown (uint32 Bank)**
Power down the EEPROM.

12.202.1 Function Documentation**12.202.1.1 CFE_PSP_EepromPowerDown()** `int32 CFE_PSP_EepromPowerDown (`
`uint32 Bank)`

Power down the EEPROM.

Parameters

in	<i>Bank</i>	The bank of EEPROM to power down
----	-------------	----------------------------------

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.2 CFE_PSP_EepromPowerUp() `int32 CFE_PSP_EepromPowerUp (`
`uint32 Bank)`

Power up the EEPROM.

Parameters

in	<i>Bank</i>	The bank of EEPROM to power up
----	-------------	--------------------------------

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.3 CFE_PSP_EepromWrite16() `int32 CFE_PSP_EepromWrite16 (`
`cpuaddr MemoryAddress,`
`uint16 uint16Value)`

Write two bytes (*uint16Value*) to EEPROM address *MemoryAddress*.

Parameters

out	<i>MemoryAddress</i>	Memory address to write to
in	<i>uint16Value</i>	Value to write to memory

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.4 CFE_PSP_EepromWrite32() `int32 CFE_PSP_EepromWrite32 (`
`cpuaddr MemoryAddress,`
`uint32 uint32Value)`

Write four bytes (*uint32Value*) to EEPROM address *MemoryAddress*.

Parameters

out	<i>MemoryAddress</i>	Memory address to write to
in	<i>uint32Value</i>	Value to write to memory

Return values

<i>CFE_PSP_SUCCESS</i>	on success
------------------------	------------

Return values

<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.5 CFE_PSP_EepromWrite8() `int32 CFE_PSP_EepromWrite8 (`

```
cpuaddr MemoryAddress,  
uint8 ByteValue )
```

Write one byte (ByteValue) to EEPROM address MemoryAddress.

Parameters

<i>out</i>	<i>MemoryAddress</i>	Memory address to write to
<i>in</i>	<i>ByteValue</i>	Value to write to memory

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.6 CFE_PSP_EepromWriteDisable() `int32 CFE_PSP_EepromWriteDisable (`

```
uint32 Bank )
```

Disable the EEPROM from write operation.

Parameters

<i>in</i>	<i>Bank</i>	The bank of EEPROM to disable
-----------	-------------	-------------------------------

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.202.1.7 CFE_PSP_EepromWriteEnable() `int32 CFE_PSP_EepromWriteEnable (`

```
uint32 Bank )
```

Enable the EEPROM for write operation.

Parameters

<i>in</i>	<i>Bank</i>	The bank of EEPROM to enable
-----------	-------------	------------------------------

Return values

<code>CFE_PSP_SUCCESS</code>	on success
<code>CFE_PSP_ERROR_NOT_IMPLEMENTED</code>	if not implemented

12.203 psp/fsw/inc/cfe_psp_error.h File Reference

cFE PSP Error header

```
#include "common_types.h"
```

Macros

- `#define CFE_PSP_STATUS_C(X) ((CFE_PSP_Status_t)(X))`
PSP Status macro for literal.
- `#define CFE_PSP_STATUS_STRING_LENGTH 12`
PSP Status converted to string length limit.
- `#define CFE_PSP_SUCCESS (CFE_PSP_STATUS_C(0))`
- `#define CFE_PSP_ERROR (CFE_PSP_STATUS_C(-1))`
- `#define CFE_PSP_INVALID_POINTER (CFE_PSP_STATUS_C(-2))`
- `#define CFE_PSP_ERROR_ADDRESS_MISALIGNED (CFE_PSP_STATUS_C(-3))`
- `#define CFE_PSP_ERROR_TIMEOUT (CFE_PSP_STATUS_C(-4))`
- `#define CFE_PSP_INVALID_INT_NUM (CFE_PSP_STATUS_C(-5))`
- `#define CFE_PSP_INVALID_MEM_ADDR (CFE_PSP_STATUS_C(-21))`
- `#define CFE_PSP_INVALID_MEM_TYPE (CFE_PSP_STATUS_C(-22))`
- `#define CFE_PSP_INVALID_MEM_RANGE (CFE_PSP_STATUS_C(-23))`
- `#define CFE_PSP_INVALID_MEM_WORDSIZE (CFE_PSP_STATUS_C(-24))`
- `#define CFE_PSP_INVALID_MEM_SIZE (CFE_PSP_STATUS_C(-25))`
- `#define CFE_PSP_INVALID_MEM_ATTR (CFE_PSP_STATUS_C(-26))`
- `#define CFE_PSP_ERROR_NOT_IMPLEMENTED (CFE_PSP_STATUS_C(-27))`
- `#define CFE_PSP_INVALID_MODULE_NAME (CFE_PSP_STATUS_C(-28))`
- `#define CFE_PSP_INVALID_MODULE_ID (CFE_PSP_STATUS_C(-29))`
- `#define CFE_PSP_NO_EXCEPTION_DATA (CFE_PSP_STATUS_C(-30))`

Typedefs

- `typedef int32 CFE_PSP_Status_t`
PSP Status type for readability and potentially type safety.
- `typedef char CFE_PSP_StatusString_t[CFE_PSP_STATUS_STRING_LENGTH]`
For the `CFE_PSP_StatusToString()` function, to ensure everyone is making an array of the same length.

Functions

- `char * CFE_PSP_StatusToString (CFE_PSP_Status_t status, CFE_PSP_StatusString_t *status_string)`
Convert status to a string.

12.203.1 Detailed Description

cFE PSP Error header

12.203.2 Macro Definition Documentation

12.203.2.1 CFE_PSP_ERROR #define CFE_PSP_ERROR (CFE_PSP_STATUS_C(-1))

Definition at line 67 of file cfe_psp_error.h.

12.203.2.2 CFE_PSP_ERROR_ADDRESS_MISALIGNED #define CFE_PSP_ERROR_ADDRESS_MISALIGNED (CFE_PSP_STATUS_C(-3))

Definition at line 69 of file cfe_psp_error.h.

12.203.2.3 CFE_PSP_ERROR_NOT_IMPLEMENTED #define CFE_PSP_ERROR_NOT_IMPLEMENTED (CFE_PSP_STATUS_C(-27))

Definition at line 78 of file cfe_psp_error.h.

12.203.2.4 CFE_PSP_ERROR_TIMEOUT #define CFE_PSP_ERROR_TIMEOUT (CFE_PSP_STATUS_C(-4))

Definition at line 70 of file cfe_psp_error.h.

12.203.2.5 CFE_PSP_INVALID_INT_NUM #define CFE_PSP_INVALID_INT_NUM (CFE_PSP_STATUS_C(-5))

Definition at line 71 of file cfe_psp_error.h.

12.203.2.6 CFE_PSP_INVALID_MEM_ADDR #define CFE_PSP_INVALID_MEM_ADDR (CFE_PSP_STATUS_C(-21))

Definition at line 72 of file cfe_psp_error.h.

12.203.2.7 CFE_PSP_INVALID_MEM_ATTR #define CFE_PSP_INVALID_MEM_ATTR (CFE_PSP_STATUS_C(-26))

Definition at line 77 of file cfe_psp_error.h.

12.203.2.8 CFE_PSP_INVALID_MEM_RANGE #define CFE_PSP_INVALID_MEM_RANGE (CFE_PSP_STATUS_C(-23))

Definition at line 74 of file cfe_psp_error.h.

12.203.2.9 CFE_PSP_INVALID_MEM_SIZE #define CFE_PSP_INVALID_MEM_SIZE (CFE_PSP_STATUS_C(-25))

Definition at line 76 of file cfe_psp_error.h.

12.203.2.10 CFE_PSP_INVALID_MEM_TYPE #define CFE_PSP_INVALID_MEM_TYPE (CFE_PSP_STATUS_C(-22))

Definition at line 73 of file cfe_psp_error.h.

12.203.2.11 CFE_PSP_INVALID_MEM_WORDSIZE #define CFE_PSP_INVALID_MEM_WORDSIZE (CFE_PSP_STATUS_C(-24))

Definition at line 75 of file cfe_psp_error.h.

12.203.2.12 CFE_PSP_INVALID_MODULE_ID #define CFE_PSP_INVALID_MODULE_ID (CFE_PSP_STATUS_C(-29))

Definition at line 80 of file cfe_psp_error.h.

12.203.2.13 CFE_PSP_INVALID_MODULE_NAME #define CFE_PSP_INVALID_MODULE_NAME (CFE_PSP_STATUS_C(-28))
Definition at line 79 of file cfe_psp_error.h.

12.203.2.14 CFE_PSP_INVALID_POINTER #define CFE_PSP_INVALID_POINTER (CFE_PSP_STATUS_C(-2))
Definition at line 68 of file cfe_psp_error.h.

12.203.2.15 CFE_PSP_NO_EXCEPTION_DATA #define CFE_PSP_NO_EXCEPTION_DATA (CFE_PSP_STATUS_C(-30))
Definition at line 81 of file cfe_psp_error.h.

12.203.2.16 CFE_PSP_STATUS_C #define CFE_PSP_STATUS_C(

X) ((CFE_PSP_Status_t) (X))

PSP Status macro for literal.

Definition at line 37 of file cfe_psp_error.h.

12.203.2.17 CFE_PSP_STATUS_STRING_LENGTH #define CFE_PSP_STATUS_STRING_LENGTH 12

PSP Status converted to string length limit.

Used for sizing CFE_PSP_StatusString_t intended for use in printing CFE_PSP_Status_t values Sized for Id (LONG←_MIN) including NULL

Definition at line 45 of file cfe_psp_error.h.

12.203.2.18 CFE_PSP_SUCCESS #define CFE_PSP_SUCCESS (CFE_PSP_STATUS_C(0))

Definition at line 66 of file cfe_psp_error.h.

12.203.3 Typedef Documentation

12.203.3.1 CFE_PSP_Status_t typedef int32 CFE_PSP_Status_t

PSP Status type for readability and potentially type safety.

Definition at line 32 of file cfe_psp_error.h.

12.203.3.2 CFE_PSP_StatusString_t typedef char CFE_PSP_StatusString_t[CFE_PSP_STATUS_STRING_LENGTH]

For the CFE_PSP_StatusToString() function, to ensure everyone is making an array of the same length.

Definition at line 51 of file cfe_psp_error.h.

12.203.4 Function Documentation

12.203.4.1 CFE_PSP_StatusToString() char* CFE_PSP_StatusToString (

```
CFE_PSP_Status_t status,
CFE_PSP_StatusString_t * status_string )
```

Convert status to a string.

Parameters

in	status	Status value to convert
----	--------	-------------------------

Parameters

<code>out</code>	<code>status_string</code>	Buffer to store status converted to string
------------------	----------------------------	--

Returns

Passed in string pointer

12.204 psp/fsw/inc/cfe_psp_exception_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- void [CFE_PSP_AttachExceptions](#) (void)

Sets up the exception environment for the chosen platform.
- void [CFE_PSP_SetDefaultExceptionEnvironment](#) (void)

Defines the CPU and FPU exceptions that are enabled for each cFE Task/App.
- uint32 [CFE_PSP_Exception_GetCount](#) (void)

Returns the unread exception count.
- int32 [CFE_PSP_Exception_GetSummary](#) (uint32 *ContextLogId, [osal_id_t](#) *TaskId, char *ReasonBuf, uint32 ReasonSize)

Retrieves a summary of an exception log entry.
- int32 [CFE_PSP_Exception_CopyContext](#) (uint32 ContextLogId, void *ContextBuf, uint32 ContextSize)

Retrieves exception log entry context information.

12.204.1 Function Documentation**12.204.1.1 CFE_PSP_AttachExceptions()** `void CFE_PSP_AttachExceptions (`
`void)`

Sets up the exception environment for the chosen platform.

On a board, this can be configured to look at a debug flag or switch in order to keep the standard OS exception handlers, rather than restarting the system.

12.204.1.2 CFE_PSP_Exception_CopyContext() `int32 CFE_PSP_Exception_CopyContext (`
`uint32 ContextLogId,`
`void * ContextBuf,`
`uint32 ContextSize)`

Retrieves exception log entry context information.

Parameters

<code>in</code>	<code>ContextLogId</code>	ID of the exception log entry to copy
<code>out</code>	<code>ContextBuf</code>	Pointer to the buffer where the context information is to be copied to
<code>in</code>	<code>ContextSize</code>	Maximum size of context information data to copy

Returns

Size of the copied data

Return values

<code>CFE_PSP_NO_EXCEPTION_DATA</code>	if data has expired from the memory log
--	---

12.204.1.3 CFE_PSP_Exception_GetCount() `uint32 CFE_PSP_Exception_GetCount (void)`

Returns the unread exception count.

Returns

The unread exception count

12.204.1.4 CFE_PSP_Exception_GetSummary() `int32 CFE_PSP_Exception_GetSummary (uint32 * ContextLogId, osal_id_t * TaskId, char * ReasonBuf, uint32 ReasonSize)`

Retrieves a summary of an exception log entry.

Note

This function returns CFE_PSP_SUCCESS to indicate that an entry was popped from the queue. This doesn't necessarily mean that the output fields have valid data, but it does mean they are initialized to something.

Parameters

in	<i>ContextLogId</i>	ID of the exception log entry to get a summary for
in,out	<i>TaskId</i>	Pointer to the TaskID buffer
out	<i>ReasonBuf</i>	Pointer to the buffer that will store the exception summary string
in	<i>ReasonSize</i>	Maximum size of the summary string to retrieve

Return values

<code>CFE_PSP_SUCCESS</code>	on success (see note above)
<code>CFE_PSP_NO_EXCEPTION_DATA</code>	if no context available for reading

12.204.1.5 CFE_PSP_SetDefaultExceptionEnvironment() `void CFE_PSP_SetDefaultExceptionEnvironment (void)`

Defines the CPU and FPU exceptions that are enabled for each cFE Task/App.

This function sets a default exception environment that can be used

Note

The exception environment is local to each task. Therefore, this must be Called for each task that wants to do floating point and catch exceptions.

12.205 psp/fsw/inc/cfe_psp_id_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- `uint32 CFE_PSP_GetProcessorId (void)`
Returns the CPU ID as defined by the specific board and BSP.
- `uint32 CFE_PSP_GetSpacecraftId (void)`
Returns the Spacecraft ID (if any)
- `const char * CFE_PSP_GetProcessorName (void)`
Returns the processor name.

12.205.1 Function Documentation**12.205.1.1 CFE_PSP_GetProcessorId()** `uint32 CFE_PSP_GetProcessorId (`
`void)`

Returns the CPU ID as defined by the specific board and BSP.

Returns

The processor ID

12.205.1.2 CFE_PSP_GetProcessorName() `const char* CFE_PSP_GetProcessorName (`
`void)`

Returns the processor name.

Returns

The processor name

12.205.1.3 CFE_PSP_GetSpacecraftId() `uint32 CFE_PSP_GetSpacecraftId (`
`void)`

Returns the Spacecraft ID (if any)

Returns

The Spacecraft ID

12.206 psp/fsw/inc/cfe_psp_memaccess_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- `int32 CFE_PSP_MemRead8 (cpuaddr MemoryAddress, uint8 *ByteValue)`
Read one byte of memory.
- `int32 CFE_PSP_MemWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)`
Write one byte of memory.
- `int32 CFE_PSP_MemRead16 (cpuaddr MemoryAddress, uint16 *uint16Value)`
Read 2 bytes of memory.
- `int32 CFE_PSP_MemWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)`
Write 2 bytes of memory.
- `int32 CFE_PSP_MemRead32 (cpuaddr MemoryAddress, uint32 *uint32Value)`
Read 4 bytes of memory.
- `int32 CFE_PSP_MemWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)`
Write 4 bytes of memory.
- `int32 CFE_PSP_MemCpy (void *dest, const void *src, uint32 n)`
Copy 'n' bytes from 'src' to 'dest'.
- `int32 CFE_PSP_MemSet (void *dest, uint8 value, uint32 n)`
Copy 'n' bytes of value 'value' to 'dest'.

12.206.1 Function Documentation

12.206.1.1 CFE_PSP_MemCpy() `int32 CFE_PSP_MemCpy (`
 `void * dest,`
 `const void * src,`
 `uint32 n)`

Copy 'n' bytes from 'src' to 'dest'.

Copies 'n' bytes from memory address pointed to by 'src' to memory address pointed to by 'dest'.

Note

For now we are using the standard C library call 'memcpy' but if we find we need to make it more efficient then we'll implement it in assembly.

Parameters

<code>out</code>	<code>dest</code>	Pointer to the destination address to copy to
<code>in</code>	<code>src</code>	Pointer to the address to copy from
<code>in</code>	<code>n</code>	Number of bytes to copy

Returns

Always returns CFE_PSP_SUCCESS

12.206.1.2 CFE_PSP_MemRead16() `int32 CFE_PSP_MemRead16 (`
 `cpuaddr MemoryAddress,`
 `uint16 * uint16Value)`

Read 2 bytes of memory.

Parameters

in	<i>MemoryAddress</i>	Address to be read
out	<i>uint16Value</i>	The address content will be copied to the location pointed to by this argument

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.206.1.3 CFE_PSP_MemRead32() `int32 CFE_PSP_MemRead32 (`

```
    cpuaddr MemoryAddress,
    uint32 * uint32Value )
```

Read 4 bytes of memory.

Parameters

in	<i>MemoryAddress</i>	Address to be read
out	<i>uint32Value</i>	The address content will be copied to the location pointed to by this argument

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.206.1.4 CFE_PSP_MemRead8() `int32 CFE_PSP_MemRead8 (`

```
    cpuaddr MemoryAddress,
    uint8 * ByteValue )
```

Read one byte of memory.

Parameters

in	<i>MemoryAddress</i>	Address to be read
out	<i>ByteValue</i>	The address content will be copied to the location pointed to by this argument

Returns

Always returns CFE_PSP_SUCCESS (if implemented)

Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

```
12.206.1.5 CFE_PSP_MemSet() int32 CFE_PSP_MemSet (
    void * dest,
    uint8 value,
    uint32 n )
```

Copy 'n' bytes of value 'value' to 'dest'.

Copies 'n' number of bytes of value 'value' to memory address pointed to by 'dest'.

Note

For now we are using the standard C library call 'memset' but if we find we need to make it more efficient then we'll implement it in assembly.

Parameters

out	<i>dest</i>	Pointer to the destination address to copy to
in	<i>value</i>	Value to set
in	<i>n</i>	Number of bytes to copy

Returns

Always returns CFE_PSP_SUCCESS

```
12.206.1.6 CFE_PSP_MemWrite16() int32 CFE_PSP_MemWrite16 (
    cpuaddr MemoryAddress,
    uint16 uint16Value )
```

Write 2 bytes of memory.

Parameters

out	<i>MemoryAddress</i>	Address to be written to
in	<i>uint16Value</i>	The content pointed to by this argument will be copied to the address

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

```
12.206.1.7 CFE_PSP_MemWrite32() int32 CFE_PSP_MemWrite32 (
    cpuaddr MemoryAddress,
    uint32 uint32Value )
```

Write 4 bytes of memory.

Parameters

out	<i>MemoryAddress</i>	Address to be written to
-----	----------------------	--------------------------

Parameters

in	<i>uint32Value</i>	The content pointed to by this argument will be copied to the address
----	--------------------	---

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.206.1.8 CFE_PSP_MemWrite8() `int32 CFE_PSP_MemWrite8 (`
`cpuaddr MemoryAddress,`
`uint8 ByteValue)`

Write one byte of memory.

Parameters

out	<i>MemoryAddress</i>	Address to be written to
in	<i>ByteValue</i>	The content pointed to by this argument will be copied to the address

Returns

Always returns CFE_PSP_SUCCESS (if implemented)

Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

12.207 psp/fsw/inc/cfe_psp_memrange_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Macros

- #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

Functions

- `int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)`
Returns the location and size of the ES Reset information area.
- `int32 CFE_PSP.GetUserReservedArea (cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)`
Returns the location and size of the memory used for the cFE user-reserved area.
- `int32 CFE_PSP_GetVolatileDiskMem (cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)`
Returns the location and size of the memory used for the cFE volatile disk.
- `int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)`
Returns the location and size of the kernel memory.
- `int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)`
Returns the location and size of the kernel memory.
- `int32 CFE_PSP_MemValidateRange (cpuaddr Address, size_t Size, uint32 MemoryType)`
Validates the memory range and type using the global CFE_PSP_MemoryTable.
- `uint32 CFE_PSP_MemRanges (void)`
Returns the number of memory ranges in the CFE_PSP_MemoryTable.
- `int32 CFE_PSP_MemRangeSet (uint32 RangeNum, uint32 MemoryType, cpuaddr StartAddr, size_t Size, size_t WordSize, uint32 Attributes)`
Populates one of the records in the CFE_PSP_MemoryTable.
- `int32 CFE_PSP_MemRangeGet (uint32 RangeNum, uint32 *MemoryType, cpuaddr *StartAddr, size_t *Size, size_t *WordSize, uint32 *Attributes)`
Retrieves one of the records in the CFE_PSP_MemoryTable.

12.207.1 Macro Definition Documentation

12.207.1.1 CFE_PSP_MEM_ANY `#define CFE_PSP_MEM_ANY 3`
Definition at line 53 of file cfe_psp_memrange_api.h.

12.207.1.2 CFE_PSP_MEM_ATTR_READ `#define CFE_PSP_MEM_ATTR_READ 0x02`
Definition at line 60 of file cfe_psp_memrange_api.h.

12.207.1.3 CFE_PSP_MEM_ATTR_READWRITE `#define CFE_PSP_MEM_ATTR_READWRITE 0x03`
Definition at line 61 of file cfe_psp_memrange_api.h.

12.207.1.4 CFE_PSP_MEM_ATTR_WRITE `#define CFE_PSP_MEM_ATTR_WRITE 0x01`
Definition at line 59 of file cfe_psp_memrange_api.h.

12.207.1.5 CFE_PSP_MEM_EEPROM `#define CFE_PSP_MEM_EEPROM 2`
Definition at line 52 of file cfe_psp_memrange_api.h.

12.207.1.6 CFE_PSP_MEM_INVALID `#define CFE_PSP_MEM_INVALID 4`
Definition at line 54 of file cfe_psp_memrange_api.h.

12.207.1.7 CFE_PSP_MEM_RAM #define CFE_PSP_MEM_RAM 1
 Definition at line 51 of file cfe_psp_memrange_api.h.

12.207.1.8 CFE_PSP_MEM_SIZE_BYTE #define CFE_PSP_MEM_SIZE_BYTE 0x01
 Definition at line 66 of file cfe_psp_memrange_api.h.

12.207.1.9 CFE_PSP_MEM_SIZE_DWORD #define CFE_PSP_MEM_SIZE_DWORD 0x04
 Definition at line 68 of file cfe_psp_memrange_api.h.

12.207.1.10 CFE_PSP_MEM_SIZE_WORD #define CFE_PSP_MEM_SIZE_WORD 0x02
 Definition at line 67 of file cfe_psp_memrange_api.h.

12.207.2 Function Documentation

12.207.2.1 CFE_PSP_GetCFETextSegmentInfo() int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr * PtrToCFEsegment,
 uint32 * SizeOfCFEsegment)

Returns the location and size of the kernel memory.

This function returns the start and end address of the CFE text segment. It may not be implemented on all architectures.

Parameters

out	<i>PtrToCFEsegment</i>	Pointer to the variable that will store the location of the cFE text segment
out	<i>SizeOfCFEsegment</i>	Pointer to the variable that will store the size of the cFE text segment

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.207.2.2 CFE_PSP_GetKernelTextSegmentInfo() int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr * PtrToKernelSegment,
 uint32 * SizeOfKernelSegment)

Returns the location and size of the kernel memory.

This function returns the start and end address of the kernel text segment. It may not be implemented on all architectures.

Parameters

out	<i>PtrToKernelSegment</i>	Pointer to the variable that will store the location of the kernel text segment
out	<i>SizeOfKernelSegment</i>	Pointer to the variable that will store the size of the kernel text segment

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error or CFE_PSP_ERROR_NOT_IMPLEMENTED if not implemented

12.207.2.3 CFE_PSP_GetResetArea() `int32 CFE_PSP_GetResetArea (`
 `cpuaddr * PtrToResetArea,`
 `uint32 * SizeOfResetArea)`

Returns the location and size of the ES Reset information area.

This area is preserved during a processor reset and is used to store the ER Log, System Log and reset-related variables

Parameters

<code>out</code>	<code>PtrToResetArea</code>	Pointer to the variable that will store the location of the reset area
<code>out</code>	<code>SizeOfResetArea</code>	Pointer to the variable that will store the reset area size

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.207.2.4 CFE_PSP.GetUserReservedArea() `int32 CFE_PSP.GetUserReservedArea (`
 `cpuaddr * PtrToUserArea,`
 `uint32 * SizeOfUserArea)`

Returns the location and size of the memory used for the cFE user-reserved area.

Parameters

<code>out</code>	<code>PtrToUserArea</code>	Pointer to the variable that will store the location of the user-reserved area
<code>out</code>	<code>SizeOfUserArea</code>	Pointer to the variable that will store the size of the user-reserved area

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

12.207.2.5 CFE_PSP_GetVolatileDiskMem() `int32 CFE_PSP_GetVolatileDiskMem (`
 `cpuaddr * PtrToVolDisk,`
 `uint32 * SizeOfVolDisk)`

Returns the location and size of the memory used for the cFE volatile disk.

Parameters

<code>out</code>	<code>PtrToVolDisk</code>	Pointer to the variable that will store the location of the cFE volatile disk
<code>out</code>	<code>SizeOfVolDisk</code>	Pointer to the variable that will store the size of the cFE volatile disk

Returns

0 (OS_SUCCESS or CFE_PSP_SUCCESS) on success, -1 (OS_ERROR or CFE_PSP_ERROR) on error

```
12.207.2.6 CFE_PSP_MemRangeGet() int32 CFE_PSP_MemRangeGet (
    uint32 RangeNum,
    uint32 * MemoryType,
    cpuaddr * StartAddr,
    size_t * Size,
    size_t * WordSize,
    uint32 * Attributes )
```

Retrieves one of the records in the CFE_PSP_MemoryTable.

Note

Because the table is fixed size, the entries are accessed by using the integer index.

Parameters

in	<i>RangeNum</i>	A 32-bit integer (starting with 0) specifying the MemoryTable entry.
out	<i>MemoryType</i>	A pointer to the 32-bit integer where the Memory Type is stored. Any defined CFE_PSP_MEM_* enumeration can be specified
out	<i>StartAddr</i>	Pointer to the 32-bit integer where the 32-bit starting address of the memory range is stored.
out	<i>Size</i>	A pointer to the 32-bit integer where the 32-bit size of the memory range is stored.
out	<i>WordSize</i>	A pointer to the 32-bit integer where the minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD)
out	<i>Attributes</i>	The attributes of the Memory Range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE)

Return values

<i>CFE_PSP_SUCCESS</i>	Memory range returned successfully.
<i>CFE_PSP_INVALID_POINTER</i>	Parameter error
<i>CFE_PSP_INVALID_MEM_RANGE</i>	The index into the table is invalid

```
12.207.2.7 CFE_PSP_MemRanges() uint32 CFE_PSP_MemRanges (
    void )
```

Returns the number of memory ranges in the CFE_PSP_MemoryTable.

Returns

Positive integer number of entries in the memory range table

```
12.207.2.8 CFE_PSP_MemRangeSet() int32 CFE_PSP_MemRangeSet (
    uint32 RangeNum,
    uint32 MemoryType,
    cpuaddr StartAddr,
    size_t Size,
    size_t WordSize,
    uint32 Attributes )
```

Populates one of the records in the CFE_PSP_MemoryTable.

Note

Because the table is fixed size, the entries are set by using the integer index. No validation is done with the address or size.

Parameters

in	<i>RangeNum</i>	A 32-bit integer (starting with 0) specifying the MemoryTable entry.
in	<i>MemoryType</i>	The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY Any defined CFE_PSP_MEM_* enumeration can be specified
in	<i>StartAddr</i>	A 32-bit starting address of the memory range
in	<i>Size</i>	A 32-bit size of the memory range (Address + Size = End Address)
in	<i>WordSize</i>	The minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD)
in	<i>Attributes</i>	The attributes of the Memory Range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE)

Return values

<i>CFE_PSP_SUCCESS</i>	Memory range set successfully.
<i>CFE_PSP_INVALID_MEM_RANGE</i>	The index into the table is invalid
<i>FE_PSP_INVALID_MEM_ADDR</i>	Starting address is not valid
<i>CFE_PSP_INVALID_MEM_TYPE</i>	Memory type associated with the range does not match the passed-in type.
<i>CFE_PSP_INVALID_MEM_WORDSIZE</i>	The WordSize parameter is not one of the predefined types.
<i>CFE_PSP_INVALID_MEM_ATTR</i>	The Attributes parameter is not one of the predefined types.
<i>OP_INVALID_MEM_SIZE</i>	The Memory range associated with the address is not large enough to contain Address + Size.

12.207.2.9 CFE_PSP_MemValidateRange() `int32 CFE_PSP_MemValidateRange (`

```
    cpuaddr Address,
    size_t Size,
    uint32 MemoryType )
```

Validates the memory range and type using the global CFE_PSP_MemoryTable.

Parameters

in	<i>Address</i>	A 32-bit starting address of the memory range
in	<i>Size</i>	A 32-bit size of the memory range (Address + Size = End Address)
in	<i>MemoryType</i>	The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY Any defined CFE_PSP_MEM_* enumeration can be specified

Return values

<i>CFE_PSP_SUCCESS</i>	Memory range and type information is valid and can be used.
------------------------	---

Return values

<code>CFE_PSP_INVALID_MEM_ADDR</code>	Starting address is not valid
<code>CFE_PSP_INVALID_MEM_TYPE</code>	Memory type associated with the range does not match the passed-in type.
<code>CFE_PSP_INVALID_MEM_RANGE</code>	The Memory range associated with the address is not large enough to contain Address + Size.

12.208 psp/fsw/inc/cfe_psp_port_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- `int32 CFE_PSP_PortRead8 (cpuaddr PortAddress, uint8 *ByteValue)`
Read one byte of memory.
- `int32 CFE_PSP_PortWrite8 (cpuaddr PortAddress, uint8 ByteValue)`
Write one byte of memory.
- `int32 CFE_PSP_PortRead16 (cpuaddr PortAddress, uint16 *uint16Value)`
Read 2 bytes of memory.
- `int32 CFE_PSP_PortWrite16 (cpuaddr PortAddress, uint16 uint16Value)`
Write 2 bytes of memory.
- `int32 CFE_PSP_PortRead32 (cpuaddr PortAddress, uint32 *uint32Value)`
Read 4 bytes of memory.
- `int32 CFE_PSP_PortWrite32 (cpuaddr PortAddress, uint32 uint32Value)`
Write 4 bytes of memory.

12.208.1 Function Documentation

12.208.1.1 CFE_PSP_PortRead16() `int32 CFE_PSP_PortRead16 (`
`cpuaddr PortAddress,`
`uint16 * uint16Value)`

Read 2 bytes of memory.

Parameters

<code>in</code>	<code>PortAddress</code>	Address to be read
<code>out</code>	<code>uint16Value</code>	The address content will be copied to the location pointed to by this argument

Return values

<code>CFE_PSP_SUCCESS</code>	on success
<code>CFE_PSP_ERROR_ADDRESS_MISALIGNED</code>	if the address is not aligned to a 16-bit addressing scheme.
<code>CFE_PSP_ERROR_NOT_IMPLEMENTED</code>	if not implemented

```
12.208.1.2 CFE_PSP_PortRead32() int32 CFE_PSP_PortRead32 (
    cpuaddr PortAddress,
    uint32 * uint32Value )
```

Read 4 bytes of memory.

Parameters

in	<i>PortAddress</i>	Address to be read
out	<i>uint32Value</i>	The address content will be copied to the location pointed to by this argument

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

```
12.208.1.3 CFE_PSP_PortRead8() int32 CFE_PSP_PortRead8 (
    cpuaddr PortAddress,
    uint8 * ByteValue )
```

Read one byte of memory.

Parameters

in	<i>PortAddress</i>	Address to be read
out	<i>ByteValue</i>	The address content will be copied to the location pointed to by this argument

Returns

Always returns *CFE_PSP_SUCCESS* (if implemented)

Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

```
12.208.1.4 CFE_PSP_PortWrite16() int32 CFE_PSP_PortWrite16 (
    cpuaddr PortAddress,
    uint16 uint16Value )
```

Write 2 bytes of memory.

Parameters

out	<i>PortAddress</i>	Address to be written to
in	<i>uint16Value</i>	the content pointed to by this argument will be copied to the address

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.208.1.5 CFE_PSP_PortWrite32() `int32 CFE_PSP_PortWrite32 (`
 `cpuaddr PortAddress,`
 `uint32 uint32Value)`

Write 4 bytes of memory.

Parameters

<i>out</i>	<i>PortAddress</i>	Address to be written to
<i>in</i>	<i>uint32Value</i>	The content pointed to by this argument will be copied to the address

Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

12.208.1.6 CFE_PSP_PortWrite8() `int32 CFE_PSP_PortWrite8 (`
 `cpuaddr PortAddress,`
 `uint8 ByteValue)`

Write one byte of memory.

Parameters

<i>out</i>	<i>PortAddress</i>	Address to be written to
<i>in</i>	<i>ByteValue</i>	The content pointed to by this argument will be copied to the address

Returns

Always returns CFE_PSP_SUCCESS (if implemented)

Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

12.209 psp/fsw/inc/cfe_psp_ssr_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
```

```
#include "cfe_psp_error.h"
```

Functions

- `int32 CFE_PSP_InitSSR (uint32 bus, uint32 device, char *DeviceName)`

Initializes the Solid State recorder memory for a particular platform.

12.209.1 Function Documentation

12.209.1.1 CFE_PSP_InitSSR() `int32 CFE_PSP_InitSSR (`

```
    uint32 bus,
    uint32 device,
    char * DeviceName )
```

Initializes the Solid State recorder memory for a particular platform.

Note

For the MCP750, this simply initializes the Hard Disk device.

Parameters

in	<i>bus</i>	
in	<i>device</i>	
in	<i>DeviceName</i>	

Return values

<code>CFE_PSP_SUCCESS</code>	on success
<code>CFE_PSP_ERROR</code>	on error

12.210 psp/fsw/inc/cfe_psp_timertick_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Macros

- `#define CFE_PSP_SOFT_TIMEBASE_NAME "cFS-Master"`

The name of the software/RTOS timebase for general system timers.

Functions

- `void CFE_PSP_GetTime (OS_time_t *LocalTime)`
Sample/Read a monotonic platform clock with normalization.
- `uint32 CFE_PSP_GetTimerTicksPerSecond (void)`
- `uint32 CFE_PSP_GetTimerLow32Rollover (void)`

- void [CFE_PSP_Get_Timebase](#) (uint32 *Tbu, uint32 *Tbl)
Sample/Read a monotonic platform clock without normalization.

12.210.1 Macro Definition Documentation

12.210.1.1 CFE_PSP_SOFT_TIMEBASE_NAME #define CFE_PSP_SOFT_TIMEBASE_NAME "cFS-Master"

The name of the software/RTOS timebase for general system timers.

This name may be referred to by CFE TIME and/or SCH when setting up its own timers.

Definition at line 53 of file cfe_psp_timertick_api.h.

12.210.2 Function Documentation

12.210.2.1 CFE_PSP_Get_Timebase() void CFE_PSP_Get_Timebase (

```
    uint32 * Tbu,
    uint32 * Tbl )
```

Sample/Read a monotonic platform clock without normalization.

Provides a common interface to system timebase. This routine is in the BSP because it is sometimes implemented in hardware and sometimes taken care of by the RTOS.

This is defined as a free-running, monotonically-increasing tick counter. The epoch is not defined, but typically is the system boot time, and the value increases indefinitely as the system runs. The tick period/rate is also not defined.

Rollover events - where the range of representable values is exceeded - are theoretically possible, but would take many years of continuous uptime to occur (typically hundreds of years, if not thousands). System designers should ensure that the actual tick rate and resulting timebase range is sufficiently large to ensure that rollover is not a concern.

Note

This is a "raw" value from the underlying platform with minimal/no conversions or normalization applied. Neither the epoch nor the resolution of this tick counter is specified, and it may vary from platform to platform. Use the [CFE_PSP_GetTime\(\)](#) function to sample the timebase and also convert the units into a normalized/more consistent form.

See also

[CFE_PSP_GetTime\(\)](#)

Parameters

out	Tbu	Buffer to hold the upper 32 bits of a 64-bit tick counter
out	Tbl	Buffer to hold the lower 32 bits of a 64-bit tick counter

12.210.2.2 CFE_PSP_GetTime() void CFE_PSP_GetTime (

```
    OS_time_t * LocalTime )
```

Sample/Read a monotonic platform clock with normalization.

Outputs an [OS_time_t](#) value indicating the time elapsed since an epoch. The epoch is not defined, but typically represents the system boot time. The value increases continuously over time and cannot be reset by software.

This is similar to the [CFE_PSP_Get_Timebase\(\)](#), but additionally it normalizes the output value to an [OS_time_t](#), thereby providing consistent units to the calling application. Any OSAL-provided routine that accepts [OS_time_t](#) inputs may be

used to convert this value into other standardized time units.

Note

This should refer to the same time domain as [CFE_PSP_Get_Timebase\(\)](#), the primary difference being the format and units of the output value.

See also

[CFE_PSP_Get_Timebase\(\)](#)

Parameters

<code>out</code>	<code>LocalTime</code>	Value of PSP tick counter as OS_time_t
------------------	------------------------	--

12.210.2.3 CFE_PSP_GetTimerLow32Rollover() `uint32 CFE_PSP_GetTimerLow32Rollover (void)`

Provides the number that the least significant 32 bits of the 64-bit time stamp returned by CFE_PSP_Get_Timebase rolls over. If the lower 32 bits rolls at 1 second, then the CFE_PSP_TIMER_LOW32_ROLLOVER will be 1000000. If the lower 32 bits rolls at its maximum value (2^{32}) then CFE_PSP_TIMER_LOW32_ROLLOVER will be 0.

Returns

The number that the least significant 32 bits of the 64-bit time stamp returned by CFE_PSP_Get_Timebase rolls over.

12.210.2.4 CFE_PSP_GetTimerTicksPerSecond() `uint32 CFE_PSP_GetTimerTicksPerSecond (void)`

Provides the resolution of the least significant 32 bits of the 64-bit time stamp returned by CFE_PSP_Get_Timebase in timer ticks per second. The timer resolution for accuracy should not be any slower than 1000000 ticks per second or 1 us (microsecond) per tick

Returns

The number of timer ticks per second of the time stamp returned by CFE_PSP_Get_Timebase

12.211 psp/fsw/inc/cfe_psp_version_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

Functions

- `const char * CFE_PSP_GetVersionString (void)`
Obtain the PSP version/baseline identifier string.
- `const char * CFE_PSP_GetVersionCodeName (void)`
Obtain the version code name.
- `void CFE_PSP_GetVersionNumber (uint8 VersionNumbers[4])`
Retrieves the numeric PSP version identifier as an array of uint8 values.
- `uint32 CFE_PSP_GetBuildNumber (void)`
Obtain the PSP library numeric build number.

12.211.1 Function Documentation

12.211.1.1 CFE_PSP_GetBuildNumber() `uint32 CFE_PSP_GetBuildNumber (void)`

Obtain the PSP library numeric build number.

The build number is a monotonically increasing number that (coarsely) reflects the number of commits/changes that have been merged since the epoch release. During development cycles this number should increase after each subsequent merge/modification.

Like other version information, this is a fixed number assigned at compile time.

Returns

The OSAL library build number

12.211.1.2 CFE_PSP_GetVersionCodeName() `const char* CFE_PSP_GetVersionCodeName (void)`

Obtain the version code name.

This retrieves the PSP code name. This is a compatibility indicator for the overall NASA cFS ecosystem. All modular components which are intended to interoperate should report the same code name.

Returns

Code name. This is a fixed string and cannot be NULL.

12.211.1.3 CFE_PSP_GetVersionNumber() `void CFE_PSP_GetVersionNumber (uint8 VersionNumbers[4])`

Retrieves the numeric PSP version identifier as an array of uint8 values.

The array of numeric values is in order of precedence: [0] = Major Number [1] = Minor Number [2] = Revision Number [3] = Mission Revision

The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. 0 indicates an official release 1-254 local patch level (reserved for mission use) 255 indicates a development build

Parameters

<code>out</code>	<code>VersionNumbers</code>	A fixed-size array to be filled with the version numbers
------------------	-----------------------------	--

12.211.1.4 CFE_PSP_GetVersionString() `const char* CFE_PSP_GetVersionString (void)`

Obtain the PSP version/baseline identifier string.

This retrieves the PSP version identifier string without extra info

Returns

Version string. This is a fixed string and cannot be NULL.

12.212 psp/fsw/inc/cfe_psp_watchdog_api.h File Reference

```
#include "common_types.h"
```

```
#include "osapi.h"
#include "cfe_psp_error.h"
```

Macros

Definitions for PSP PANIC types

- #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

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.
- #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_Restart](#) (uint32 resetType)
Entry point back to the BSP to restart the processor.
- uint32 [CFE_PSP_GetRestartType](#) (uint32 *restartSubType)
Returns the last reset type.
- void [CFE_PSP_WatchdogInit](#) (void)
Configures the watchdog timer.
- void [CFE_PSP_WatchdogEnable](#) (void)
Enables the watchdog timer.

- void **CFE_PSP_WatchdogDisable** (void)
Disables the watchdog timer.
- void **CFE_PSP_WatchdogService** (void)
Services the watchdog timer according to the value set in WatchDogSet.
- uint32 **CFE_PSP_WatchdogGet** (void)
Gets the watchdog time in milliseconds.
- void **CFE_PSP_WatchdogSet** (uint32 WatchdogValue)
Sets the watchdog time in milliseconds.
- void **CFE_PSP_Panic** (int32 ErrorCode)
Aborts the cFE startup.

12.212.1 Macro Definition Documentation

12.212.1.1 CFE_PSP_PANIC_CORE_APP #define CFE_PSP_PANIC_CORE_APP 6
Definition at line 59 of file cfe_psp_watchdog_api.h.

12.212.1.2 CFE_PSP_PANIC_GENERAL_FAILURE #define CFE_PSP_PANIC_GENERAL_FAILURE 7
Definition at line 60 of file cfe_psp_watchdog_api.h.

12.212.1.3 CFE_PSP_PANIC_MEMORY_ALLOC #define CFE_PSP_PANIC_MEMORY_ALLOC 3
Definition at line 56 of file cfe_psp_watchdog_api.h.

12.212.1.4 CFE_PSP_PANIC_NONVOL_DISK #define CFE_PSP_PANIC_NONVOL_DISK 4
Definition at line 57 of file cfe_psp_watchdog_api.h.

12.212.1.5 CFE_PSP_PANIC_STARTUP #define CFE_PSP_PANIC_STARTUP 1
Definition at line 54 of file cfe_psp_watchdog_api.h.

12.212.1.6 CFE_PSP_PANIC_STARTUP_SEM #define CFE_PSP_PANIC_STARTUP_SEM 5
Definition at line 58 of file cfe_psp_watchdog_api.h.

12.212.1.7 CFE_PSP_PANIC_VOLATILE_DISK #define CFE_PSP_PANIC_VOLATILE_DISK 2
Definition at line 55 of file cfe_psp_watchdog_api.h.

12.212.1.8 CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET #define CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET 9
Reset reverted to a cFE POWERON due to a boot bank switch.
Definition at line 96 of file cfe_psp_watchdog_api.h.

12.212.1.9 CFE_PSP_RST_SUBTYPE_EXCEPTION #define CFE_PSP_RST_SUBTYPE_EXCEPTION 6

Reset was caused by a Processor Exception.

Definition at line 90 of file cfe_psp_watchdog_api.h.

12.212.1.10 CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND #define CFE_PSP_RST_SUBTYPE_HW_SPEC←

IAL_COMMAND 3

Reset was caused by a reset line having been stimulated by a hardware special command.

Definition at line 84 of file cfe_psp_watchdog_api.h.

12.212.1.11 CFE_PSP_RST_SUBTYPE_HW_WATCHDOG #define CFE_PSP_RST_SUBTYPE_HW_WATCHDOG 4

Reset was caused by a watchdog timer expiring.

Definition at line 86 of file cfe_psp_watchdog_api.h.

12.212.1.12 CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET #define CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET 8

Reset was caused by a JTAG or BDM connection.

Definition at line 94 of file cfe_psp_watchdog_api.h.

12.212.1.13 CFE_PSP_RST_SUBTYPE_MAX #define CFE_PSP_RST_SUBTYPE_MAX 10

Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Definition at line 98 of file cfe_psp_watchdog_api.h.

12.212.1.14 CFE_PSP_RST_SUBTYPE_POWER_CYCLE #define CFE_PSP_RST_SUBTYPE_POWER_CYCLE 1

Reset caused by power having been removed and restored.

Definition at line 80 of file cfe_psp_watchdog_api.h.

12.212.1.15 CFE_PSP_RST_SUBTYPE_PUSH_BUTTON #define CFE_PSP_RST_SUBTYPE_PUSH_BUTTON 2

Reset caused by reset button on the board.

Definition at line 82 of file cfe_psp_watchdog_api.h.

12.212.1.16 CFE_PSP_RST_SUBTYPE_RESET_COMMAND #define CFE_PSP_RST_SUBTYPE_RESET_COMMAND 5

Reset was caused by cFE ES processing a [Reset Command](#).

Definition at line 88 of file cfe_psp_watchdog_api.h.

12.212.1.17 CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET #define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7

SET 7

Reset was caused in an unknown manner.

Definition at line 92 of file cfe_psp_watchdog_api.h.

12.212.1.18 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 70 of file cfe_psp_watchdog_api.h.

12.212.1.19 CFE_PSP_RST_TYPE_POWERON #define CFE_PSP_RST_TYPE_POWERON 2
 All memory has been cleared
 Definition at line 69 of file cfe_psp_watchdog_api.h.

12.212.1.20 CFE_PSP_RST_TYPE_PROCESSOR #define CFE_PSP_RST_TYPE_PROCESSOR 1
 Volatile disk, CDS and User Reserved memory may be valid
 Definition at line 68 of file cfe_psp_watchdog_api.h.

12.212.2 Function Documentation

12.212.2.1 CFE_PSP_GetRestartType() uint32 CFE_PSP_GetRestartType (uint32 * restartSubType)

Returns the last reset type.

Note

If a pointer to a valid memory space is passed in, it returns the reset sub-type in that memory. Right now the reset types are application-specific. For the cFE they are defined in the [cfe_es.h](#) file.

Parameters

restartSubType	
----------------	--

12.212.2.2 CFE_PSP_Panic() void CFE_PSP_Panic (int32 ErrorCode)

Aborts the cFE startup.

Provides a common interface to abort the cFE startup process and return back to the OS.

Note

This is called by the cFE Core startup code when it needs to abort the cFE startup. This should not be called by applications.

Parameters

in	ErrorCode	Reason for exiting
----	-----------	--------------------

12.212.2.3 CFE_PSP_Restart() void CFE_PSP_Restart (uint32 resetType)

Entry point back to the BSP to restart the processor.

The flight software calls this routine to restart the processor.

Parameters

in	resetType	Type of reset
----	-----------	---------------

```
12.212.2.4 CFE_PSP_WatchdogDisable() void CFE_PSP_WatchdogDisable (
    void )
```

Disables the watchdog timer.

```
12.212.2.5 CFE_PSP_WatchdogEnable() void CFE_PSP_WatchdogEnable (
    void )
```

Enables the watchdog timer.

```
12.212.2.6 CFE_PSP_WatchdogGet() uint32 CFE_PSP_WatchdogGet (
    void )
```

Gets the watchdog time in milliseconds.

Returns

The current watchdog value

```
12.212.2.7 CFE_PSP_WatchdogInit() void CFE_PSP_WatchdogInit (
    void )
```

Configures the watchdog timer.

To set up the timer resolution and/or other settings custom to this platform.

```
12.212.2.8 CFE_PSP_WatchdogService() void CFE_PSP_WatchdogService (
    void )
```

Services the watchdog timer according to the value set in WatchDogSet.

Load the watchdog timer with a count that corresponds to the millisecond time given in the parameter.

Note

Currently an ExpireTime value of zero will result in the minimum reset time of 4.5 seconds. All other ExpireTime values will result in a reset time of 5.5 seconds.

```
12.212.2.9 CFE_PSP_WatchdogSet() void CFE_PSP_WatchdogSet (
    uint32 WatchdogValue )
```

Sets the watchdog time in milliseconds.

Parameters

in	<i>WatchdogValue</i>	New watchdog value to set
----	----------------------	---------------------------

Index

EXTENSION
 common_types.h, [1699](#)

accuracy
 OS_timebase_prop_t, [827](#)
 OS_timer_prop_t, [828](#)

ack
 CF_Logical_IntHeader, [611](#)

ack_directive_code
 CF_Logical_PduAck, [613](#)

ack_limit
 CF_ChannelConfig, [575](#)
 CF_HkFault, [604](#)

ack_subtype_code
 CF_Logical_PduAck, [613](#)

ack_timer
 CF_Transaction, [644](#)

ack_timer_armed
 CF_Flags_Common, [592](#)

ack_timer_s
 CF_ChannelConfig, [575](#)

acknak_count
 CF_RxS2_Data, [636](#)
 CF_TxS2_Data, [654](#)

action
 CF_ChанAction_SuspResArg, [572](#)

ActiveBuffer
 CFE_TBL_HousekeepingTlm_Payload, [770](#)

ActiveBufferAddr
 CFE_TBL_TblRegPacket_Payload, [781](#)

ActiveTableFlag
 CFE_TBL_DumpCmd_Payload, [764](#)
 CFE_TBL_ValidateCmd_Payload, [784](#)

ActualLength
 OS_SockAddr_t, [822](#)

ActualType
 CFE_Config_ValueEntry, [660](#)

addr
 OS_module_prop_t, [820](#)

AddrData
 OS_SockAddr_t, [822](#)

Address
 OS_static_symbol_record_t, [824](#)

AddressesAreValid
 CFE_ES_AppInfo, [662](#)

AlignPtr
 OS_SockAddrData_t, [823](#)

AlignU32
 OS_SockAddrData_t, [823](#)

AppData
 CFE_EVS_HousekeepingTlm_Payload, [717](#)

AppDataFilename
 CFE_EVS_AppDataCmd_Payload, [705](#)

AppEnableStatus
 CFE_EVS_AppTlmData, [708](#)

AppEntryPoint
 CFE_ES_StartAppCmd_Payload, [698](#)

AppFileName
 CFE_ES_AppReloadCmd_Payload, [666](#)
 CFE_ES_StartAppCmd_Payload, [698](#)

AppID
 CFE_EVS_AppTlmData, [708](#)

AppId
 CFE_ES_TaskInfo, [702](#)
 CFE_SB_PipeInfoEntry, [746](#)

AppInfo
 CFE_ES_OneAppTlm_Payload, [684](#)

Application
 CFE_ES_AppNameCmd_Payload, [665](#)
 CFE_ES_AppReloadCmd_Payload, [666](#)
 CFE_ES_SendMemPoolStatsCmd_Payload, [693](#)
 CFE_ES_StartAppCmd_Payload, [698](#)

ApplicationID
 CFE_FS_Header, [733](#)

AppMessageSentCounter
 CFE_EVS_AppTlmData, [708](#)

AppMessageSquelchedCounter
 CFE_EVS_AppTlmData, [709](#)

AppName
 CFE_ES_TaskInfo, [702](#)
 CFE_EVS_AppNameBitMaskCmd_Payload, [706](#)
 CFE_EVS_AppNameCmd_Payload, [706](#)
 CFE_EVS_AppNameEventIDCmd_Payload, [707](#)
 CFE_EVS_AppNameEventIDMaskCmd_Payload, [707](#)
 CFE_EVS_PacketID, [723](#)
 CFE_SB_PipeInfoEntry, [746](#)
 CFE_SB_RoutingFileEntry, [749](#)

apps/cf/config/default_cf_extern_typedefs.h, [828](#)

apps/cf/config/default_cf_fcnCodes.h, [831](#)

apps/cf/config/default_cf_interface_cfg.h, [831](#)

apps/cf/config/default_cf_internal_cfg.h, [832](#)

apps/cf/config/default_cf_mission_cfg.h, [833](#)

apps/cf/config/default_cf_msg.h, [833](#)

apps/cf/config/default_cf_msgdefs.h, [834](#)

apps/cf/config/default_cf_msgIds.h, [836](#)

apps/cf/config/default_cf_msgStruct.h, [836](#)

apps/cf/config/default_cf_platform_cfg.h, [839](#)

apps/cf/config/default_cf_tbl.h, [840](#)

apps/cf/config/default_cf_tblDefs.h, [840](#)

apps/cf/config/default_cf_tblStruct.h, [841](#)

apps/cf/config/default_cf_topicIds.h, [841](#)

apps/cf/docs/dox_src/cfs_cf.dox, 843
 apps/cf/fsw/inc(cf_events.h, 843
 apps/cf/fsw/inc(cf_perfids.h, 848
 apps/cf/fsw/src(cf_app.c, 849
 apps/cf/fsw/src(cf_app.h, 855
 apps/cf/fsw/src(cf_assert.h, 862
 apps/cf/fsw/src(cf_cfdp.c, 863
 apps/cf/fsw/src(cf_cfdp.h, 907
 apps/cf/fsw/src(cf_cfdp_dispatch.c, 947
 apps/cf/fsw/src(cf_cfdp_dispatch.h, 951
 apps/cf/fsw/src(cf_cfdp_pdu.h, 955
 apps/cf/fsw/src(cf_cfdp_r.c, 961
 apps/cf/fsw/src(cf_cfdp_r.h, 985
 apps/cf/fsw/src(cf_cfdp_s.c, 1008
 apps/cf/fsw/src(cf_cfdp_s.h, 1029
 apps/cf/fsw/src(cf_cfdp_sbintf.c, 1049
 apps/cf/fsw/src(cf_cfdp_sbintf.h, 1053
 apps/cf/fsw/src(cf_cfdp_types.h, 1058
 apps/cf/fsw/src(cf_chunk.c, 1065
 apps/cf/fsw/src(cf_chunk.h, 1075
 apps/cf/fsw/src(cf_clist.c, 1086
 apps/cf/fsw/src(cf_clist.h, 1091
 apps/cf/fsw/src(cf_cmd.c, 1097
 apps/cf/fsw/src(cf_cmd.h, 1129
 apps/cf/fsw/src(cf_codec.c, 1162
 apps/cf/fsw/src(cf_codec.h, 1191
 apps/cf/fsw/src(cf_crc.c, 1220
 apps/cf/fsw/src(cf_crc.h, 1222
 apps/cf/fsw/src(cf_dispatch.c, 1224
 apps/cf/fsw/src(cf_dispatch.h, 1226
 apps/cf/fsw/src(cf_eds_dispatch.c, 1228
 apps/cf/fsw/src(cf_logical_pdu.h, 1231
 apps/cf/fsw/src(cf_timer.c, 1235
 apps/cf/fsw/src(cf_timer.h, 1237
 apps/cf/fsw/src(cf_utils.c, 1240
 apps/cf/fsw/src(cf_utils.h, 1258
 apps/cf/fsw/src(cf_verify.h, 1280
 apps/cf/fsw/src(cf_version.h, 1280
 apps/cf/fsw/tables(cf_def_config.c, 1281
ARGCHECK
 osapi-macros.h, 1721
AsInteger
 CFE_Config_ValueBuffer, 660
AsPointer
 CFE_Config_ValueBuffer, 660
AtToneDelay
 CFE_TIME_DiagnosticTlm_Payload, 789
AtToneLatch
 CFE_TIME_DiagnosticTlm_Payload, 789
AtToneLeapSeconds
 CFE_TIME_DiagnosticTlm_Payload, 789
 CFE_TIME_ToneDataCmd_Payload, 812
AtToneMET
 CFE_TIME_DiagnosticTlm_Payload, 789
 CFE_TIME_ToneDataCmd_Payload, 812
AtToneState
 CFE_TIME_ToneDataCmd_Payload, 812
AtToneSTCF
 CFE_TIME_DiagnosticTlm_Payload, 789
 CFE_TIME_ToneDataCmd_Payload, 812
barg
 CF_ChanAction_BoolArg, 571
 CF_ChanAction_BoolMsgArg, 571
base
 CF_DecoderState, 583
 CF_EncoderState, 588
BitMask
 CFE_EVS_AppNameBitMaskCmd_Payload, 706
 CFE_EVS_BitMaskCmd_Payload, 710
block_size
 OS_statvfs_t, 825
blocks_free
 OS_statvfs_t, 825
BlockSize
 CFE_ES_BlockStats, 666
BlockStats
 CFE_ES_MemPoolStats, 681
BootSource
 CFE_ES_HousekeepingTlm_Payload, 675
bss_address
 OS_module_address_t, 819
bss_size
 OS_module_address_t, 819
BSSAddress
 CFE_ES_AppInfo, 662
BSSSize
 CFE_ES_AppInfo, 662
Buffer
 OS_SockAddrData_t, 823
BUGCHECK
 osapi-macros.h, 1721
BUGCHECK_VOID
 osapi-macros.h, 1722
BUGREPORT
 osapi-macros.h, 1722
buildosal_public_api/inc/osconfig.h, 1281
busy
 CF_Playback, 629
byte
 CF_UnionArgs_Payload, 656
ByteAlign4
 CFE_TBL_TblRegPacket_Payload, 781
ByteAlignPad1
 CFE_TBL_HousekeepingTlm_Payload, 770
ByteAlignSpare
 CFE_ES_CDSRegDumpRec, 667
cached_pos

CF_RxState_Data, 637
CF_TxState_Data, 655
canceled
 CF_Flags_Common, 593
cc
 CF_CFDP_PduEof, 559
 CF_Logical_PduAck, 613
 CF_Logical_PduEof, 615
 CF_Logical_PduFin, 617
cc_and_transaction_status
 CF_CFDP_PduAck, 559
CCSDS_ExtendedHeader, 553
 Subsystem, 553
 SystemId, 553
CCSDS_ExtendedHeader_t
 ccsds_hdr.h, 1552
ccsds_hdr.h
 CCSDS_ExtendedHeader_t, 1552
 CCSDS_PrimaryHeader_t, 1553
CCSDS_PrimaryHeader, 553
 Length, 554
 Sequence, 554
 StreamId, 554
CCSDS_PrimaryHeader_t
 ccsds_hdr.h, 1553
CdsName
 CFE_ES_DeleteCDSCmd_Payload, 670
CF_ABANDON_CC
 CFS CFDP Command Codes, 205
CF_Abandon_TxnCmd
 cf_cmd.c, 1099
 cf_cmd.h, 1132
CF_AbandonCmd, 554
 cf_cmd.c, 1100
 cf_cmd.h, 1133
 CommandHeader, 554
 Payload, 554
CF_AbandonCmd_t
 CFS CFDP Command Structures, 226
CF_ALL_CHANNELS
 default_cf_msg.h, 834
CF_ALL_POLLDIRS
 default_cf_msg.h, 834
cf_app.c
 CF_AppData, 855
 CF_ApplInit, 850
 CF_AppMain, 851
 CF_CheckTables, 852
 CF_TableInit, 853
 CF_ValidateConfigTable, 854
cf_app.h
 CF_AppData, 862
 CF_ApplInit, 857
 CF_AppMain, 858
CF_CHANNEL_PIPE_PREFIX, 856
CF_CheckTables, 859
CF_ERROR, 857
CF_PDU_METADATA_ERROR, 857
CF_PIPE_NAME, 857
CF_REC_PDU_BAD_EOF_ERROR, 857
CF_REC_PDUFSIZE_MISMATCH_ERROR, 857
CF_SEND_PDU_ERROR, 857
CF_SEND_PDU_NO_BUF_AVAIL_ERROR, 857
CF_SHORT_PDU_ERROR, 857
CF_TableInit, 860
CF_ValidateConfigTable, 861
CF_APP_MAX_HEADER_SIZE
 cf_cfdp_pdu.h, 957
CF_AppData
 cf_app.c, 855
 cf_app.h, 862
CF_AppData_t, 555
 CmdPipe, 555
 config_handle, 555
 config_table, 555
 engine, 555
 hk, 556
 RunStatus, 556
CF_ApplInit
 cf_app.c, 850
 cf_app.h, 857
CF_AppMain
 cf_app.c, 851
 cf_app.h, 858
CF_AppPipe
 cf_dispatch.c, 1224
 cf_dispatch.h, 1226
 cf_eds_dispatch.c, 1229
CF_Assert
 cf_assert.h, 863
cf_assert.h
 CF_Assert, 863
CF_CANCEL_CC
 CFS CFDP Command Codes, 204
CF_Cancel_TxnCmd
 cf_cmd.c, 1101
 cf_cmd.h, 1134
CF_CancelCmd, 556
 cf_cmd.c, 1102
 cf_cmd.h, 1135
 CommandHeader, 556
 Payload, 556
CF_CancelCmd_t
 CFS CFDP Command Structures, 227
CF_CC_ERR_EID
 CFS CFDP Event IDs, 158
cf_cfdp.c
 CF_CFDP_AppendTlv, 866

CF_CFDP_ArmAckTimer, 866
 CF_CFDP_ArmInactTimer, 867
 CF_CFDP_CancelTransaction, 867
 CF_CFDP_CloseFiles, 868
 CF_CFDP_ConstructPduHeader, 869
 CF_CFDP_CopyStringFromLV, 870
 CF_CFDP_CycleEngine, 871
 CF_CFDP_CycleTx, 871
 CF_CFDP_CycleTxFirstActive, 872
 CF_CFDP_DecodeStart, 873
 CF_CFDP_DisableEngine, 874
 CF_CFDP_DispatchRecv, 875
 CF_CFDP_DispatchTx, 876
 CF_CFDP_DoTick, 877
 CF_CFDP_EncodeStart, 878
 CF_CFDP_FindUnusedChunks, 878
 CF_CFDP_GetClass, 879
 CF_CFDP_HandleNotKeepFile, 879
 CF_CFDP_InitEngine, 879
 CF_CFDP_InitTxnTxFile, 881
 CF_CFDP_IsPollingDir, 881
 CF_CFDP_IsSender, 882
 CF_CFDP_MoveFile, 882
 CF_CFDP_PlaybackDir, 883
 CF_CFDP_PlaybackDir_Initiate, 884
 CF_CFDP_ProcessPlaybackDirectories, 885
 CF_CFDP_ProcessPlaybackDirectory, 885
 CF_CFDP_ProcessPollingDirectories, 886
 CF_CFDP_RecvAck, 887
 CF_CFDP_RecvDrop, 888
 CF_CFDP_RecvEof, 888
 CF_CFDP_RecvFd, 889
 CF_CFDP_RecvFin, 890
 CF_CFDP_RecvIdle, 891
 CF_CFDP_RecvMd, 892
 CF_CFDP_RecvNak, 893
 CF_CFDP_RecvPh, 894
 CF_CFDP_ResetTransaction, 895
 CF_CFDP_SendAck, 896
 CF_CFDP_SendEof, 897
 CF_CFDP_SendEotPkt, 898
 CF_CFDP_SendFd, 899
 CF_CFDP_SendFin, 900
 CF_CFDP_SendMd, 901
 CF_CFDP_SendNak, 902
 CF_CFDP_SetPduLength, 903
 CF_CFDP_SetTxnStatus, 903
 CF_CFDP_TickTransactions, 904
 CF_CFDP_TxFile, 905
 CF_CFDP_TxFile_Initiate, 906
 CF_CFDP_UpdatePollPbCounted, 907
 cf_cfdp.h
 CF_CFDP_AppendTlv, 910
 CF_CFDP_ArmAckTimer, 910
 CF_CFDP_CancelTransaction, 911
 CF_CFDP_CloseFiles, 912
 CF_CFDP_ConstructPduHeader, 913
 CF_CFDP_CopyStringFromLV, 914
 CF_CFDP_CycleEngine, 915
 CF_CFDP_CycleTx, 915
 CF_CFDP_CycleTx_args_t, 910
 CF_CFDP_CycleTxFirstActive, 916
 CF_CFDP_DecodeStart, 917
 CF_CFDP_DisableEngine, 918
 CF_CFDP_DispatchRecv, 919
 CF_CFDP_DoTick, 920
 CF_CFDP_EncodeStart, 921
 CF_CFDP_HandleNotKeepFile, 921
 CF_CFDP_InitEngine, 922
 CF_CFDP_InitTxnTxFile, 923
 CF_CFDP_IsPollingDir, 924
 CF_CFDP_MoveFile, 924
 CF_CFDP_PlaybackDir, 925
 CF_CFDP_ProcessPlaybackDirectory, 926
 CF_CFDP_ProcessPollingDirectories, 927
 CF_CFDP_RecvAck, 928
 CF_CFDP_RecvDrop, 929
 CF_CFDP_RecvEof, 929
 CF_CFDP_RecvFd, 930
 CF_CFDP_RecvFin, 931
 CF_CFDP_RecvIdle, 932
 CF_CFDP_RecvMd, 933
 CF_CFDP_RecvNak, 934
 CF_CFDP_RecvPh, 935
 CF_CFDP_ResetTransaction, 936
 CF_CFDP_SendAck, 937
 CF_CFDP_SendEof, 938
 CF_CFDP_SendEotPkt, 939
 CF_CFDP_SendFd, 940
 CF_CFDP_SendFin, 941
 CF_CFDP_SendMd, 942
 CF_CFDP_SendNak, 943
 CF_CFDP_SetTxnStatus, 944
 CF_CFDP_Tick_args_t, 910
 CF_CFDP_TickTransactions, 945
 CF_CFDP_TxFile, 946
 CF_CFDP_AckTxnStatus_ACTIVE
 cf_cfdp_pdu.h, 959
 CF_CFDP_AckTxnStatus_INVALID
 cf_cfdp_pdu.h, 959
 CF_CFDP_AckTxnStatus_t
 cf_cfdp_pdu.h, 959
 CF_CFDP_AckTxnStatus_TERMINATED
 cf_cfdp_pdu.h, 959
 CF_CFDP_AckTxnStatus_UNDEFINED
 cf_cfdp_pdu.h, 959
 CF_CFDP_AckTxnStatus_UNRECOGNIZED
 cf_cfdp_pdu.h, 959

CF_CFDP_AppendTlv
 cf_cfdp.c, 866
 cf_cfdp.h, 910
CF_CFDP_ArmAckTimer
 cf_cfdp.c, 866
 cf_cfdp.h, 910
CF_CFDP_ArmInactTimer
 cf_cfdp.c, 867
CF_CFDP_CancelTransaction
 cf_cfdp.c, 867
 cf_cfdp.h, 911
CF_CFDP_CLASS_1
 default_cf_extern_typedefs.h, 830
CF_CFDP_CLASS_2
 default_cf_extern_typedefs.h, 830
CF_CFDP_Class_t
 default_cf_extern_typedefs.h, 830
CF_CFDP_CLOSE_ERR_EID
 CFS CFDP Event IDs, 158
CF_CFDP_CloseFiles
 cf_cfdp.c, 868
 cf_cfdp.h, 912
CF_CFDP_CodecCheckSize
 cf_codec.c, 1165
 cf_codec.h, 1196
CF_CFDP_CodecGetPosition
 cf_codec.h, 1197
CF_CFDP_CodecGetRemain
 cf_codec.h, 1198
CF_CFDP_CodecGetSize
 cf_codec.h, 1198
CF_CFDP_CodeclsOK
 cf_codec.h, 1198
CF_CFDP_CodecReset
 cf_codec.h, 1199
CF_CFDP_CodecSetDone
 cf_codec.h, 1199
CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_FILE_SIZE_ERROR
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_FILESTORE_REJECTION
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_INACTIVITY_DETECTED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED
 cf_cfdp_pdu.h, 960
 cf_cfdp.h, 910
CF_CFDP_ConditionCode_NAK_LIMIT_REACHED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_NO_ERROR
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED
 cf_cfdp_pdu.h, 960
CF_CFDP_ConditionCode_t
 cf_cfdp_pdu.h, 959
CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE
 cf_cfdp_pdu.h, 960
CF_CFDP_ConstructPduHeader
 cf_cfdp.c, 869
 cf_cfdp.h, 913
CF_CFDP_CopyStringFromLV
 cf_cfdp.c, 870
 cf_cfdp.h, 914
CF_CFDP_CycleEngine
 cf_cfdp.c, 871
 cf_cfdp.h, 915
CF_CFDP_CycleTx
 cf_cfdp.c, 871
 cf_cfdp.h, 915
CF_CFDP_CycleTx_args, 557
 chan, 557
 ran_one, 557
CF_CFDP_CycleTx_args_t
 cf_cfdp.h, 910
CF_CFDP_CycleTxFirstActive
 cf_cfdp.c, 872
 cf_cfdp.h, 916
CF_CFDP_DecodeAck
 cf_codec.c, 1166
 cf_codec.h, 1199
CF_CFDP_DecodeAllSegments
 cf_codec.c, 1167
 cf_codec.h, 1200
CF_CFDP_DecodeAllTlv
 cf_codec.c, 1167
 cf_codec.h, 1200
CF_CFDP_DecodeCrc
 cf_codec.c, 1168
 cf_codec.h, 1201
CF_CFDP_DecodeEof
 cf_codec.c, 1168
 cf_codec.h, 1201
CF_CFDP_DecodeFileDataHeader
 cf_codec.c, 1169
 cf_codec.h, 1202
CF_CFDP_DecodeFileDirectiveHeader
 cf_codec.c, 1170
 cf_codec.h, 1202

CF_CFDP_DecodeFin
 cf_codec.c, 1170
 cf_codec.h, 1203

CF_CFDP_DecodeHeader
 cf_codec.c, 1171
 cf_codec.h, 1204

CF_CFDP_DecodeLV
 cf_codec.c, 1172
 cf_codec.h, 1205

CF_CFDP_DecodeMd
 cf_codec.c, 1172
 cf_codec.h, 1205

CF_CFDP_DecodeNak
 cf_codec.c, 1173
 cf_codec.h, 1206

CF_CFDP_DecodeSegmentRequest
 cf_codec.c, 1173
 cf_codec.h, 1206

CF_CFDP_DecodeStart
 cf_cfdp.c, 873
 cf_cfdp.h, 917

CF_CFDP_DecodeTLV
 cf_codec.c, 1174
 cf_codec.h, 1207

CF_CFDP_DIR_SLOT_ERR_EID
 CFS CFDP Event IDs, 158

CF_CFDP_DisableEngine
 cf_cfdp.c, 874
 cf_cfdp.h, 918

cf_cfdp_dispatch.c
 CF_CFDP_R_DispatchRecv, 948
 CF_CFDP_RxStateDispatch, 949
 CF_CFDP_S_DispatchRecv, 949
 CF_CFDP_S_DispatchTransmit, 950
 CF_CFDP_TxStateDispatch, 950

cf_cfdp_dispatch.h
 CF_CFDP_R_DispatchRecv, 952
 CF_CFDP_RxStateDispatch, 953
 CF_CFDP_S_DispatchRecv, 953
 CF_CFDP_S_DispatchTransmit, 954
 CF_CFDP_StateRecvFunc_t, 951
 CF_CFDP_StateSendFunc_t, 952
 CF_CFDP_TxStateDispatch, 954

CF_CFDP_DispatchRecv
 cf_cfdp.c, 875
 cf_cfdp.h, 919

CF_CFDP_DispatchTx
 cf_cfdp.c, 876

CF_CFDP_DoDecodeChunk
 cf_codec.c, 1174
 cf_codec.h, 1207

CF_CFDP_DoEncodeChunk
 cf_codec.c, 1175
 cf_codec.h, 1208

CF_CFDP_DoTick
 cf_cfdp.c, 877
 cf_cfdp.h, 920

CF_CFDP_EncodeAck
 cf_codec.c, 1176
 cf_codec.h, 1209

CF_CFDP_EncodeAllSegments
 cf_codec.c, 1176
 cf_codec.h, 1209

CF_CFDP_EncodeAllTlv
 cf_codec.c, 1177
 cf_codec.h, 1210

CF_CFDP_EncodeCrc
 cf_codec.c, 1177
 cf_codec.h, 1210

CF_CFDP_EncodeEof
 cf_codec.c, 1178
 cf_codec.h, 1211

CF_CFDP_EncodeFileDataHeader
 cf_codec.c, 1179
 cf_codec.h, 1212

CF_CFDP_EncodeFileDirectiveHeader
 cf_codec.c, 1179
 cf_codec.h, 1212

CF_CFDP_EncodeFin
 cf_codec.c, 1180
 cf_codec.h, 1213

CF_CFDP_EncodeHeaderFinalSize
 cf_codec.c, 1181
 cf_codec.h, 1214

CF_CFDP_EncodeHeaderWithoutSize
 cf_codec.c, 1181
 cf_codec.h, 1214

CF_CFDP_EncodeLV
 cf_codec.c, 1182
 cf_codec.h, 1215

CF_CFDP_EncodeMd
 cf_codec.c, 1183
 cf_codec.h, 1216

CF_CFDP_EncodeNak
 cf_codec.c, 1183
 cf_codec.h, 1216

CF_CFDP_EncodeSegmentRequest
 cf_codec.c, 1184
 cf_codec.h, 1217

CF_CFDP_EncodeStart
 cf_cfdp.c, 878
 cf_cfdp.h, 921

CF_CFDP_EncodeTlv
 cf_codec.c, 1185
 cf_codec.h, 1218

CF_CFDP_FD_UNHANDLED_ERR_EID
 CFS CFDP Event IDs, 159

CF_CFDP_FileDirective_ACK

cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_EOF
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_FIN
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_INVALID_MAX
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_INVALID_MIN
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_KEEP_ALIVE
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_METADATA
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_NAK
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_PROMPT
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirective_t
 cf_cfdp_pdu.h, 960
CF_CFDP_FileDirectiveDispatchTable_t, 557
 fdirective, 558
CF_CFDP_FinDeliveryCode_COMPLETE
 cf_cfdp_pdu.h, 960
CF_CFDP_FinDeliveryCode_INCOMPLETE
 cf_cfdp_pdu.h, 961
CF_CFDP_FinDeliveryCode_INVALID
 cf_cfdp_pdu.h, 961
CF_CFDP_FinDeliveryCode_t
 cf_cfdp_pdu.h, 960
CF_CFDP_FindUnusedChunks
 cf_cfdp.c, 878
CF_CFDP_FinFileStatus_DISCARDED
 cf_cfdp_pdu.h, 961
CF_CFDP_FinFileStatus_DISCARDED_FILESTORE
 cf_cfdp_pdu.h, 961
CF_CFDP_FinFileStatus_INVALID
 cf_cfdp_pdu.h, 961
CF_CFDP_FinFileStatus_RETAINED
 cf_cfdp_pdu.h, 961
CF_CFDP_FinFileStatus_t
 cf_cfdp_pdu.h, 961
CF_CFDP_FinFileStatus_UNREPORTED
 cf_cfdp_pdu.h, 961
CF_CFDP_GetClass
 cf_cfdp.c, 879
CF_CFDP_GetValueEncodedSize
 cf_codec.c, 1185
 cf_codec.h, 1218
CF_CFDP_HandleNotKeepFile
 cf_cfdp.c, 879
 cf_cfdp.h, 921
CF_CFDP_IDLE_MD_ERR_EID
 CFS CFDP Event IDs, 159
CF_CFDP_InitEngine
 cf_cfdp.c, 879
 cf_cfdp.h, 922
CF_CFDP_InitTxnTxFile
 cf_cfdp.c, 881
 cf_cfdp.h, 923
CF_CFDP_INVALID_DST_ERR_EID
 CFS CFDP Event IDs, 159
CF_CFDP_IsPollingDir
 cf_cfdp.c, 881
 cf_cfdp.h, 924
CF_CFDP_IsSender
 cf_cfdp.c, 882
CF_CFDP_Inv, 558
 length, 558
CF_CFDP_Inv_t
 cf_cfdp_pdu.h, 958
CF_CFDP_MAX_CMD_TX_ERR_EID
 CFS CFDP Event IDs, 159
CF_CFDP_MAX_HEADER_SIZE
 cf_cfdp_pdu.h, 957
CF_CFDP_MIN_HEADER_SIZE
 cf_cfdp_pdu.h, 958
CF_CFDP_MoveFile
 cf_cfdp.c, 882
 cf_cfdp.h, 924
CF_CFDP_MsgOutGet
 cf_cfdp_sbintf.c, 1050
 cf_cfdp_sbintf.h, 1054
CF_CFDP_NO_MSG_ERR_EID
 CFS CFDP Event IDs, 160
CF_CFDP_OPENDIR_ERR_EID
 CFS CFDP Event IDs, 160
cf_cfdp_pdu.h
 CF_APP_MAX_HEADER_SIZE, 957
 CF_CFDP_AckTxnStatus_ACTIVE, 959
 CF_CFDP_AckTxnStatus_INVALID, 959
 CF_CFDP_AckTxnStatus_t, 959
 CF_CFDP_AckTxnStatus_TERMINATED, 959
 CF_CFDP_AckTxnStatus_UNDEFINED, 959
 CF_CFDP_AckTxnStatus_UNRECOGNIZED, 959
 CF_CFDP_ConditionCode_CANCEL_REQUEST RECEIVED, 960
 CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED, 960
 CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE, 960
 CF_CFDP_ConditionCode_FILE_SIZE_ERROR, 960
 CF_CFDP_ConditionCode_FILESTORE_REJECTION, 960
 CF_CFDP_ConditionCode_INACTIVITY_DETECTED, 960
 CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE, 960

CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE, [961](#)
 CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE, [961](#)
 CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED, [961](#)
 CF_CFDP_TLV_TYPE_FLOW_LABEL, [961](#)
 CF_CFDP_TLV_TYPE_INVALID_MAX, [961](#)
 CF_CFDP_ConditionCode_NAK_LIMIT_REACHED, [961](#)
 CF_CFDP_TLV_TYPE_MESSAGE_TO_USER, [961](#)
 CF_CFDP_TlvType_t, [961](#)
 CF_CFDP_ConditionCode_NO_ERROR, [960](#)
 CF_CFDP_PduAck, [558](#)
 CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED, [960](#)
 cc_and_transaction_status, [559](#)
 directive_and_subtype_code, [559](#)
 CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED, [960](#)
 CF_CFDP_PduAck_CC
 cf_codec.c, [1189](#)
 CF_CFDP_ConditionCode_t, [959](#)
 CF_CFDP_PduAck_DIR_CODE
 cf_codec.c, [1189](#)
 CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE, [960](#)
 CF_CFDP_PduAck_DIR_SUBTYPE_CODE
 cf_codec.c, [1189](#)
 CF_CFDP_ConditionCode_t, [958](#)
 cf_cfdp_pdu.h, [958](#)
 CF_CFDP_PduAck_TRANSACTION_STATUS
 cf_codec.c, [1189](#)
 CF_CFDP_PduEof, [559](#)
 cc, [559](#)
 crc, [559](#)
 size, [560](#)
 CF_CFDP_PduEof_FLAGS_CC
 cf_codec.c, [1189](#)
 CF_CFDP_PduEof_t
 cf_cfdp_pdu.h, [958](#)
 CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE
 cf_codec.c, [1189](#)
 CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH
 cf_codec.c, [1189](#)
 CF_CFDP_PduFileContent, [560](#)
 data, [560](#)
 CF_CFDP_PduFileContent_t
 cf_cfdp_pdu.h, [958](#)
 CF_CFDP_PduFileHeader, [560](#)
 offset, [561](#)
 CF_CFDP_PduFileHeader_t
 cf_cfdp_pdu.h, [958](#)
 CF_CFDP_PduFileDirectiveHeader, [561](#)
 directive_code, [561](#)
 CF_CFDP_PduFileDirectiveHeader_t
 cf_cfdp_pdu.h, [958](#)
 CF_CFDP_PduFin, [561](#)
 flags, [562](#)
 CF_CFDP_PduFin_FLAGS_CC
 cf_codec.c, [1189](#)
 CF_CFDP_PduFin_FLAGS_DELIVERY_CODE
 cf_codec.c, [1190](#)
 CF_CFDP_PduFin_FLAGS_FILE_STATUS
 cf_codec.c, [1190](#)
 CF_CFDP_TLV_TYPE_ENTITY_ID, [961](#)
 CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE, [961](#)
 CF_CFDP_TLV_TYPE_FILESTORE_REQUEST, [961](#)
 CF_CFDP_TlvType_t, [958](#)
 CF_CFDP_PduHeader, [562](#)

eid_tsn_lengths, 562
flags, 562
length, 562
CF_CFDP_PduHeader_FLAGS_CRC
 cf_codec.c, 1190
CF_CFDP_PduHeader_FLAGS_DIR
 cf_codec.c, 1190
CF_CFDP_PduHeader_FLAGS_LARGEFILE
 cf_codec.c, 1190
CF_CFDP_PduHeader_FLAGS_MODE
 cf_codec.c, 1190
CF_CFDP_PduHeader_FLAGS_TYPE
 cf_codec.c, 1190
CF_CFDP_PduHeader_FLAGS_VERSION
 cf_codec.c, 1190
CF_CFDP_PduHeader_LENGTHHS_ENTITY
 cf_codec.c, 1191
CF_CFDP_PduHeader_LENGTHHS_TRANSACTION_SEQUENCE
 cf_codec.c, 1191
CF_CFDP_PduHeader_t
 cf_cfdp_pdu.h, 958
CF_CFDP_PduMd, 563
 segmentation_control, 563
 size, 563
CF_CFDP_PduMd_CHECKSUM_TYPE
 cf_codec.c, 1191
CF_CFDP_PduMd_CLOSURE_REQUESTED
 cf_codec.c, 1191
CF_CFDP_PduMd_t
 cf_cfdp_pdu.h, 959
CF_CFDP_PduNak, 563
 scope_end, 564
 scope_start, 564
CF_CFDP_PduNak_t
 cf_cfdp_pdu.h, 959
CF_CFDP_PlaybackDir
 cf_cfdp.c, 883
 cf_cfdp.h, 925
CF_CFDP_PlaybackDir_Initiate
 cf_cfdp.c, 884
CF_CFDP_ProcessPlaybackDirectories
 cf_cfdp.c, 885
CF_CFDP_ProcessPlaybackDirectory
 cf_cfdp.c, 885
 cf_cfdp.h, 926
CF_CFDP_ProcessPollingDirectories
 cf_cfdp.c, 886
 cf_cfdp.h, 927
cf_cfdp_r.c
 CF_CFDP_R1_Recv, 963
 CF_CFDP_R1_Reset, 964
 CF_CFDP_R1_SubstateRecvEof, 965
 CF_CFDP_R1_SubstateRecvFileData, 966
 CF_CFDP_R2_CalcCrcChunk, 967
CF_CFDP_R2_Complete, 968
CF_CFDP_R2_GapCompute, 969
CF_CFDP_R2_Recv, 969
CF_CFDP_R2_Recv_fin_ack, 970
CF_CFDP_R2_RecvMd, 971
CF_CFDP_R2_Reset, 972
CF_CFDP_R2_SetFinTxnStatus, 973
CF_CFDP_R2_SubstateRecvEof, 974
CF_CFDP_R2_SubstateRecvFileData, 975
CF_CFDP_R2_SubstateSendFin, 976
CF_CFDP_R_Cancel, 977
CF_CFDP_R_CheckCrc, 978
CF_CFDP_R_Init, 979
CF_CFDP_R_ProcessFd, 980
CF_CFDP_R_SendInactivityEvent, 981
CF_CFDP_R_SubstateRecvEof, 981
CF_CFDP_R_SubstateSendNak, 982
cf_cfdp_r.h
 CF_CFDP_R1_Recv, 987
 CF_CFDP_R1_Reset, 987
 CF_CFDP_R1_SubstateRecvEof, 988
 CF_CFDP_R1_SubstateRecvFileData, 989
 CF_CFDP_R2_CalcCrcChunk, 990
 CF_CFDP_R2_Complete, 991
 CF_CFDP_R2_GapCompute, 992
 CF_CFDP_R2_Recv, 993
 CF_CFDP_R2_Recv_fin_ack, 993
 CF_CFDP_R2_RecvMd, 994
 CF_CFDP_R2_Reset, 995
 CF_CFDP_R2_SetFinTxnStatus, 996
 CF_CFDP_R2_SubstateRecvEof, 997
 CF_CFDP_R2_SubstateRecvFileData, 998
 CF_CFDP_R2_SubstateSendFin, 999
 CF_CFDP_R_Cancel, 1000
 CF_CFDP_R_CheckCrc, 1001
 CF_CFDP_R_Init, 1002
 CF_CFDP_R_ProcessFd, 1003
 CF_CFDP_R_SendInactivityEvent, 1004
 CF_CFDP_R_SubstateRecvEof, 1004
 CF_CFDP_R_SubstateSendNak, 1005
 CF_CFDP_R_Tick, 1006
CF_CFDP_R1_Recv
 cf_cfdp_r.c, 963
 cf_cfdp_r.h, 987
CF_CFDP_R1_Reset
 cf_cfdp_r.c, 964
 cf_cfdp_r.h, 987
CF_CFDP_R1_SubstateRecvEof
 cf_cfdp_r.c, 965
 cf_cfdp_r.h, 988
CF_CFDP_R1_SubstateRecvFileData
 cf_cfdp_r.c, 966
 cf_cfdp_r.h, 989

CF_CFDP_R2_CalcCrcChunk
 cf_cfdp_r.c, 967
 cf_cfdp_r.h, 990

CF_CFDP_R2_Complete
 cf_cfdp_r.c, 968
 cf_cfdp_r.h, 991

CF_CFDP_R2_GapCompute
 cf_cfdp_r.c, 969
 cf_cfdp_r.h, 992

CF_CFDP_R2_Recv
 cf_cfdp_r.c, 969
 cf_cfdp_r.h, 993

CF_CFDP_R2_Recv_fin_ack
 cf_cfdp_r.c, 970
 cf_cfdp_r.h, 993

CF_CFDP_R2_RecvMd
 cf_cfdp_r.c, 971
 cf_cfdp_r.h, 994

CF_CFDP_R2_Reset
 cf_cfdp_r.c, 972
 cf_cfdp_r.h, 995

CF_CFDP_R2_SetFinTxnStatus
 cf_cfdp_r.c, 973
 cf_cfdp_r.h, 996

CF_CFDP_R2_SubstateRecvEof
 cf_cfdp_r.c, 974
 cf_cfdp_r.h, 997

CF_CFDP_R2_SubstateRecvFileData
 cf_cfdp_r.c, 975
 cf_cfdp_r.h, 998

CF_CFDP_R2_SubstateSendFin
 cf_cfdp_r.c, 976
 cf_cfdp_r.h, 999

CF_CFDP_R_ACK_LIMIT_ERR_EID
 CFS CFDP Event IDs, 160

CF_CFDP_R_Cancel
 cf_cfdp_r.c, 977
 cf_cfdp_r.h, 1000

CF_CFDP_R_CheckCrc
 cf_cfdp_r.c, 978
 cf_cfdp_r.h, 1001

CF_CFDP_R_CRC_ERR_EID
 CFS CFDP Event IDs, 160

CF_CFDP_R_CREAT_ERR_EID
 CFS CFDP Event IDs, 161

CF_CFDP_R_DC_INV_ERR_EID
 CFS CFDP Event IDs, 161

CF_CFDP_R_DispatchRecv
 cf_cfdp_dispatch.c, 948
 cf_cfdp_dispatch.h, 952

CF_CFDP_R_EOF_MD_SIZE_ERR_EID
 CFS CFDP Event IDs, 161

CF_CFDP_R_INACT_TIMER_ERR_EID
 CFS CFDP Event IDs, 161

CF_CFDP_R_Init
 cf_cfdp_r.c, 979
 cf_cfdp_r.h, 1002

CF_CFDP_R_NAK_LIMIT_ERR_EID
 CFS CFDP Event IDs, 162

CF_CFDP_R_OPEN_ERR_EID
 CFS CFDP Event IDs, 162

CF_CFDP_R_PDU_EOF_ERR_EID
 CFS CFDP Event IDs, 162

CF_CFDP_R_PDU_FINACK_ERR_EID
 CFS CFDP Event IDs, 162

CF_CFDP_R_PDU_MD_ERR_EID
 CFS CFDP Event IDs, 163

CF_CFDP_R_ProcessFd
 cf_cfdp_r.c, 980
 cf_cfdp_r.h, 1003

CF_CFDP_R_READ_ERR_EID
 CFS CFDP Event IDs, 163

CF_CFDP_R_RENAME_ERR_EID
 CFS CFDP Event IDs, 163

CF_CFDP_R_REQUEST_MD_INF_EID
 CFS CFDP Event IDs, 163

CF_CFDP_R_SEEK_CRC_ERR_EID
 CFS CFDP Event IDs, 164

CF_CFDP_R_SEEK_FD_ERR_EID
 CFS CFDP Event IDs, 164

CF_CFDP_R_SendInactivityEvent
 cf_cfdp_r.c, 981
 cf_cfdp_r.h, 1004

CF_CFDP_R_SIZE_MISMATCH_ERR_EID
 CFS CFDP Event IDs, 164

CF_CFDP_R_SubstateDispatchTable_t, 564
 state, 564

CF_CFDP_R_SubstateRecvEof
 cf_cfdp_r.c, 981
 cf_cfdp_r.h, 1004

CF_CFDP_R_SubstateSendNak
 cf_cfdp_r.c, 982
 cf_cfdp_r.h, 1005

CF_CFDP_R_TEMP_FILE_INF_EID
 CFS CFDP Event IDs, 164

CF_CFDP_R_Tick
 cf_cfdp_r.c, 983
 cf_cfdp_r.h, 1006

CF_CFDP_R_WRITE_ERR_EID
 CFS CFDP Event IDs, 165

CF_CFDP_ReceiveMessage
 cf_cfdp_sbintf.c, 1051
 cf_cfdp_sbintf.h, 1056

CF_CFDP_RecvAck
 cf_cfdp.c, 887
 cf_cfdp.h, 928

CF_CFDP_RecvDrop
 cf_cfdp.c, 888

cf_cfdp.h, 929
CF_CFDP_RecvEof
 cf_cfdp.c, 888
 cf_cfdp.h, 929
CF_CFDP_RecvFd
 cf_cfdp.c, 889
 cf_cfdp.h, 930
CF_CFDP_RecvFin
 cf_cfdp.c, 890
 cf_cfdp.h, 931
CF_CFDP_RecvIdle
 cf_cfdp.c, 891
 cf_cfdp.h, 932
CF_CFDP_RecvMd
 cf_cfdp.c, 892
 cf_cfdp.h, 933
CF_CFDP_RecvNak
 cf_cfdp.c, 893
 cf_cfdp.h, 934
CF_CFDP_RecvPh
 cf_cfdp.c, 894
 cf_cfdp.h, 935
CF_CFDP_ResetTransaction
 cf_cfdp.c, 895
 cf_cfdp.h, 936
CF_CFDP_RX_DROPPED_ERR_EID
 CFS CFDP Event IDs, 165
CF_CFDP_RxStateDispatch
 cf_cfdp_dispatch.c, 949
 cf_cfdp_dispatch.h, 953
cf_cfdp_s.c
 CF_CFDP_S1_Recv, 1010
 CF_CFDP_S1_SubstateSendEof, 1010
 CF_CFDP_S1_Tx, 1011
 CF_CFDP_S2_EarlyFin, 1012
 CF_CFDP_S2_Fin, 1013
 CF_CFDP_S2_Nak, 1013
 CF_CFDP_S2_Nak_Arm, 1014
 CF_CFDP_S2_Recv, 1015
 CF_CFDP_S2_SubstateSendEof, 1016
 CF_CFDP_S2_SubstateSendFileData, 1017
 CF_CFDP_S2_Tx, 1018
 CF_CFDP_S2_WaitForEofAck, 1019
 CF_CFDP_S_Cancel, 1020
 CF_CFDP_S_CheckAndRespondNak, 1020
 CF_CFDP_S_Reset, 1021
 CF_CFDP_S_SendEof, 1022
 CF_CFDP_S_SendFileData, 1023
 CF_CFDP_S_SubstateSendFileData, 1024
 CF_CFDP_S_SubstateSendFinAck, 1025
 CF_CFDP_S_SubstateSendMetadata, 1026
 CF_CFDP_S_Tick, 1027
 CF_CFDP_S_Tick_Nak, 1028
cf_cfdp_s.h
 CF_CFDP_S1_Recv, 1030
 CF_CFDP_S1_SubstateSendEof, 1031
 CF_CFDP_S1_Tx, 1032
 CF_CFDP_S2_EarlyFin, 1033
 CF_CFDP_S2_Fin, 1034
 CF_CFDP_S2_Nak, 1034
 CF_CFDP_S2_Nak_Arm, 1035
 CF_CFDP_S2_Recv, 1036
 CF_CFDP_S2_SubstateSendEof, 1037
 CF_CFDP_S2_SubstateSendFileData, 1038
 CF_CFDP_S2_Tx, 1039
 CF_CFDP_S2_WaitForEofAck, 1040
 CF_CFDP_S_Cancel, 1041
 CF_CFDP_S_CheckAndRespondNak, 1041
 CF_CFDP_S_SendEof, 1042
 CF_CFDP_S_SendFileData, 1043
 CF_CFDP_S_SubstateSendFileData, 1044
 CF_CFDP_S_SubstateSendFinAck, 1045
 CF_CFDP_S_SubstateSendMetadata, 1046
 CF_CFDP_S_Tick, 1047
 CF_CFDP_S_Tick_Nak, 1048
CF_CFDP_S1_Recv
 cf_cfdp_s.c, 1010
 cf_cfdp_s.h, 1030
CF_CFDP_S1_SubstateSendEof
 cf_cfdp_s.c, 1010
 cf_cfdp_s.h, 1031
CF_CFDP_S1_Tx
 cf_cfdp_s.c, 1011
 cf_cfdp_s.h, 1032
CF_CFDP_S2_EarlyFin
 cf_cfdp_s.c, 1012
 cf_cfdp_s.h, 1033
CF_CFDP_S2_Fin
 cf_cfdp_s.c, 1013
 cf_cfdp_s.h, 1034
CF_CFDP_S2_Nak
 cf_cfdp_s.c, 1013
 cf_cfdp_s.h, 1034
CF_CFDP_S2_Nak_Arm
 cf_cfdp_s.c, 1014
 cf_cfdp_s.h, 1035
CF_CFDP_S2_Recv
 cf_cfdp_s.c, 1015
 cf_cfdp_s.h, 1036
CF_CFDP_S2_SubstateSendEof
 cf_cfdp_s.c, 1016
 cf_cfdp_s.h, 1037
CF_CFDP_S2_SubstateSendFileData
 cf_cfdp_s.c, 1017
 cf_cfdp_s.h, 1038
CF_CFDP_S2_Tx
 cf_cfdp_s.c, 1018
 cf_cfdp_s.h, 1039

CF_CFDP_S2_WaitForEofAck
 cf_cfdp_s.c, 1019
 cf_cfdp_s.h, 1040

CF_CFDP_S_ACK_LIMIT_ERR_EID
 CFS CFDP Event IDs, 165

CF_CFDP_S_ALREADY_OPEN_ERR_EID
 CFS CFDP Event IDs, 165

CF_CFDP_S_Cancel
 cf_cfdp_s.c, 1020
 cf_cfdp_s.h, 1041

CF_CFDP_S_CheckAndRespondNak
 cf_cfdp_s.c, 1020
 cf_cfdp_s.h, 1041

CF_CFDP_S_DC_INV_ERR_EID
 CFS CFDP Event IDs, 166

CF_CFDP_S_DispatchRecv
 cf_cfdp_dispatch.c, 949
 cf_cfdp_dispatch.h, 953

CF_CFDP_S_DispatchTransmit
 cf_cfdp_dispatch.c, 950
 cf_cfdp_dispatch.h, 954

CF_CFDP_S_EARLY_FIN_ERR_EID
 CFS CFDP Event IDs, 166

CF_CFDP_S_INACT_TIMER_ERR_EID
 CFS CFDP Event IDs, 166

CF_CFDP_S_INVALID_SR_ERR_EID
 CFS CFDP Event IDs, 166

CF_CFDP_S_NON_FD_PDU_ERR_EID
 CFS CFDP Event IDs, 167

CF_CFDP_S_OPEN_ERR_EID
 CFS CFDP Event IDs, 167

CF_CFDP_S_PDU_EOF_ERR_EID
 CFS CFDP Event IDs, 167

CF_CFDP_S_PDU_NAK_ERR_EID
 CFS CFDP Event IDs, 167

CF_CFDP_S_READ_ERR_EID
 CFS CFDP Event IDs, 168

CF_CFDP_S_Reset
 cf_cfdp_s.c, 1021

CF_CFDP_S_SEEK_BEG_ERR_EID
 CFS CFDP Event IDs, 168

CF_CFDP_S_SEEK_END_ERR_EID
 CFS CFDP Event IDs, 168

CF_CFDP_S_SEEK_FD_ERR_EID
 CFS CFDP Event IDs, 168

CF_CFDP_S_SEND_FD_ERR_EID
 CFS CFDP Event IDs, 169

CF_CFDP_S_SEND_MD_ERR_EID
 CFS CFDP Event IDs, 169

CF_CFDP_S_SendEof
 cf_cfdp_s.c, 1022
 cf_cfdp_s.h, 1042

CF_CFDP_S_SendFileData
 cf_cfdp_s.c, 1023

cf_cfdp_s.h, 1043

CF_CFDP_S_START_SEND_INF_EID
 CFS CFDP Event IDs, 169

CF_CFDP_S_SubstateRecvDispatchTable_t, 564
 substate, 565

CF_CFDP_S_SubstateSendDispatchTable_t, 565
 substate, 565

CF_CFDP_S_SubstateSendFileData
 cf_cfdp_s.c, 1024
 cf_cfdp_s.h, 1044

CF_CFDP_S_SubstateSendFinAck
 cf_cfdp_s.c, 1025
 cf_cfdp_s.h, 1045

CF_CFDP_S_SubstateSendMetadata
 cf_cfdp_s.c, 1026
 cf_cfdp_s.h, 1046

CF_CFDP_S_Tick
 cf_cfdp_s.c, 1027
 cf_cfdp_s.h, 1047

CF_CFDP_S_Tick_Nak
 cf_cfdp_s.c, 1028
 cf_cfdp_s.h, 1048

cf_cfdp_sbintf.c
 CF_CFDP_MsgOutGet, 1050
 CF_CFDP_ReceiveMessage, 1051
 CF_CFDP_Send, 1052

cf_cfdp_sbintf.h
 CF_CFDP_MsgOutGet, 1054
 CF_CFDP_ReceiveMessage, 1056
 CF_CFDP_Send, 1057
 CF_PduCmdMsg_t, 1054
 CF_PduTlmMsg_t, 1054

CF_CFDP_SegmentRequest, 565
 offset_end, 566
 offset_start, 566

CF_CFDP_SegmentRequest_t
 cf_cfdp_pdu.h, 959

CF_CFDP_Send
 cf_cfdp_sbintf.c, 1052
 cf_cfdp_sbintf.h, 1057

CF_CFDP_SendAck
 cf_cfdp.c, 896
 cf_cfdp.h, 937

CF_CFDP_SendEof
 cf_cfdp.c, 897
 cf_cfdp.h, 938

CF_CFDP_SendEotPkt
 cf_cfdp.c, 898
 cf_cfdp.h, 939

CF_CFDP_SendFd
 cf_cfdp.c, 899
 cf_cfdp.h, 940

CF_CFDP_SendFin
 cf_cfdp.c, 900

cf_cfdp.h, 941
CF_CFDP_SendMd
 cf_cfdp.c, 901
 cf_cfdp.h, 942
CF_CFDP_SendNak
 cf_cfdp.c, 902
 cf_cfdp.h, 943
CF_CFDP_SetPduLength
 cf_cfdp.c, 903
CF_CFDP_SetTxnStatus
 cf_cfdp.c, 903
 cf_cfdp.h, 944
CF_CFDP_StateRecvFunc_t
 cf_cfdp_dispatch.h, 951
CF_CFDP_StateSendFunc_t
 cf_cfdp_dispatch.h, 952
CF_CFDP_Tick_args, 566
 chan, 566
 cont, 566
 early_exit, 567
 fn, 567
CF_CFDP_Tick_args_t
 cf_cfdp.h, 910
CF_CFDP_TickTransactions
 cf_cfdp.c, 904
 cf_cfdp.h, 945
CF_CFDP_tlv, 567
 length, 567
 type, 567
CF_CFDP_tlv_t
 cf_cfdp_pdu.h, 959
CF_CFDP_TLV_TYPE_ENTITY_ID
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_FILESTORE_REQUEST
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_FLOW_LABEL
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_INVALID_MAX
 cf_cfdp_pdu.h, 961
CF_CFDP_TLV_TYPE_MESSAGE_TO_USER
 cf_cfdp_pdu.h, 961
CF_CFDP_TlvType_t
 cf_cfdp_pdu.h, 961
CF_CFDP_TxFile
 cf_cfdp.c, 905
 cf_cfdp.h, 946
CF_CFDP_TxFile_Initiate
 cf_cfdp.c, 906
CF_CFDP_TxnRecvDispatchTable_t, 568
 rx, 568
CF_CFDP_TxnSendDispatchTable_t, 568
 tx, 568
CF_CFDP_TxStateDispatch
 cf_cfdp_dispatch.c, 950
 cf_cfdp_dispatch.h, 954
cf_cfdp_types.h
 CF_Channel_t, 1062
 CF_ChunkWrapper_t, 1062
 CF_Direction_NUM, 1063
 CF_Direction_RX, 1063
 CF_Direction_t, 1063
 CF_Direction_TX, 1063
 CF_Engine_t, 1062
 CF_Flags_Common_t, 1062
 CF_Flags_Rx_t, 1062
 CF_Flags_Tx_t, 1062
 CF_History_t, 1062
 CF_Input_t, 1062
 CF_NUM_CHUNKS_ALL_CHANNELS, 1061
 CF_NUM_HISTORIES, 1061
 CF_NUM_TRANSACTIONS, 1061
 CF_NUM_TRANSACTIONS_PER_CHANNEL, 1061
 CF_Output_t, 1062
 CF_Playback_t, 1062
 CF_Poll_t, 1062
 CF_RxS2_Data_t, 1063
 CF_RxState_Data_t, 1063
 CF_RxSubState_EOF, 1064
 CF_RxSubState_FILEDATA, 1064
 CF_RxSubState_NUM_STATES, 1064
 CF_RxSubState_t, 1063
 CF_RxSubState_WAIT_FOR_FIN_ACK, 1064
 CF_StateData_t, 1063
 CF_StateFlags_t, 1063
 CF_TickType_NUM_TYPES, 1064
 CF_TickType_RX, 1064
 CF_TickType_t, 1064
 CF_TickType_TXW_NAK, 1064
 CF_TickType_TXW_NORM, 1064
 CF_Transaction_t, 1063
 CF_TxnState_DROP, 1064
 CF_TxnState_IDLE, 1064
 CF_TxnState_INVALID, 1064
 CF_TxnState_R1, 1064
 CF_TxnState_R2, 1064
 CF_TxnState_S1, 1064
 CF_TxnState_S2, 1064
 CF_TxnState_t, 1064
 CF_TxnStatus_ACK_LIMIT_NO_EOF, 1065
 CF_TxnStatus_ACK_LIMIT_NO_FIN, 1065
 CF_TxnStatus_CANCEL_REQUEST_RECEIVED, 1065
 CF_TxnStatus_CHECK_LIMIT_REACHED, 1065
 CF_TxnStatus_EARLY_FIN, 1065

CF_TxnStatus_FILE_CHECKSUM_FAILURE, 1065
 CF_TxnStatus_FILE_SIZE_ERROR, 1065
 CF_TxnStatus_FILESTORE_REJECTION, 1065
 CF_TxnStatus_INACTIVITY_DETECTED, 1065
 CF_TxnStatus_INVALID_FILE_STRUCTURE, 1065
 CF_TxnStatus_INVALID_TRANSMISSION_MODE, 1065
 CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED, 1065
 CF_TxnStatus_MAX, 1065
 CF_TxnStatus_NAK_LIMIT_REACHED, 1065
 CF_TxnStatus_NAK_RESPONSE_ERROR, 1065
 CF_TxnStatus_NO_ERROR, 1065
 CF_TxnStatus_POS_ACK_LIMIT_REACHED, 1065
 CF_TxnStatus_PROTOCOL_ERROR, 1065
 CF_TxnStatus_SEND_EOF_FAILURE, 1065
 CF_TxnStatus_SUSPEND_REQUEST_RECEIVED, 1065
 CF_TxnStatus_t, 1064
 CF_TxnStatus_UNDEFINED, 1064
 CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE, CF_ChанAction_SuspResArg_t, 1065
 CF_TxS2_Data_t, 1063
 CF_TxState_Data_t, 1063
 CF_TxSubState_EOF, 1065
 CF_TxSubState_FILENODATA, 1065
 CF_TxSubState_METADATA, 1065
 CF_TxSubState_NUM_STATES, 1065
 CF_TxSubState_SEND_FIN_ACK, 1065
 CF_TxSubState_t, 1065
 CF_TxSubState_WAIT_FOR_EOF_ACK, 1065
 CF_TxSubState_WAIT_FOR_FIN, 1065
 CF_CFDP_uint16_t, 569
 octets, 569
 CF_CFDP_uint32_t, 569
 octets, 569
 CF_CFDP_uint64_t, 570
 octets, 570
 CF_CFDP_uint8_t, 570
 octets, 570
 CF_CFDP_UpdatePollPbCounted
 cf_cfdp.c, 907
 CF_CH0_RX_MID
 CFS CFDP Data Interface Message IDs, 233
 CF_CH0_TX_MID
 CFS CFDP Data Interface Message IDs, 233
 CF_CH1_RX_MID
 CFS CFDP Data Interface Message IDs, 233
 CF_CH1_TX_MID
 CFS CFDP Data Interface Message IDs, 233
 CF_ChанAction_BoolArg, 570
 barg, 571
 CF_ChанAction_BoolArg_t
 cf_cmd.h, 1131
 CF_ChанAction_BoolMsgArg, 571
 barg, 571
 data, 571
 CF_ChанAction_BoolMsgArg_t
 cf_cmd.h, 1132
 CF_ChанAction_MsgArg, 571
 data, 572
 CF_ChанAction_MsgArg_t
 cf_cmd.h, 1132
 CF_ChанAction_Status_ERROR
 cf_cmd.h, 1132
 CF_ChанAction_Status_IS_SUCCESS
 cf_cmd.h, 1136
 CF_ChанAction_Status_SUCCESS
 cf_cmd.h, 1132
 CF_ChанAction_Status_t
 cf_cmd.h, 1132
 CF_ChанAction_SuspResArg, 572
 action, 572
 same, 572
 cf_cmd.h, 1132
 CF_ChанActionFn_t
 cf_cmd.h, 1132
 CF_Channel, 573
 cs, 573
 cur, 573
 num_cmd_tx, 573
 pipe, 573
 playback, 573
 poll, 574
 qs, 574
 sem_id, 574
 tick_type, 574
 CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION
 CFS CFDP Platform Configuration, 217
 CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION
 CFS CFDP Platform Configuration, 217
 CF_CHANNEL_PIPE_PREFIX
 cf_app.h, 856
 CF_Channel_t
 cf_cfdp_types.h, 1062
 CF_ChannelConfig, 574
 ack_limit, 575
 ack_timer_s, 575
 dequeue_enabled, 575
 inactivity_timer_s, 575
 max_outgoing_messages_per_wakeup, 575
 mid_input, 575
 mid_output, 576
 move_dir, 576
 nak_limit, 576
 nak_timer_s, 576
 pipe_depth_input, 576

polldir, 576
rx_max_messages_per_wakeup, 576
sem_name, 576
CF_ChannelConfig_t
 default_cf_tbldefs.h, 840
CF_CheckTables
 cf_app.c, 852
 cf_app.h, 859
CF_Chunk, 577
 offset, 577
 size, 577
cf_chunk.c
 CF_ChunkList_ComputeGaps, 1066
 CF_ChunkList_GetFirstChunk, 1067
 CF_ChunkList_RemoveFromFirst, 1068
 CF_ChunkListAdd, 1068
 CF_ChunkListInit, 1069
 CF_ChunkListReset, 1070
 CF_Chunks_CombineNext, 1070
 CF_Chunks_CombinePrevious, 1071
 CF_Chunks_EraseChunk, 1072
 CF_Chunks_EraseRange, 1072
 CF_Chunks_FindInsertPosition, 1072
 CF_Chunks_FindSmallestSize, 1073
 CF_Chunks_Insert, 1073
 CF_Chunks_InsertChunk, 1074
cf_chunk.h
 CF_Chunk_MAX, 1077
 CF_Chunk_t, 1076
 CF_ChunkIdx_t, 1076
 CF_ChunkList_ComputeGapFn_t, 1076
 CF_ChunkList_ComputeGaps, 1077
 CF_ChunkList_GetFirstChunk, 1078
 CF_ChunkList_RemoveFromFirst, 1078
 CF_ChunkList_t, 1077
 CF_ChunkListAdd, 1079
 CF_ChunkListInit, 1080
 CF_ChunkListReset, 1080
 CF_ChunkOffset_t, 1077
 CF_Chunks_CombineNext, 1081
 CF_Chunks_CombinePrevious, 1082
 CF_Chunks_EraseChunk, 1082
 CF_Chunks_EraseRange, 1083
 CF_Chunks_FindInsertPosition, 1083
 CF_Chunks_FindSmallestSize, 1084
 CF_Chunks_Insert, 1084
 CF_Chunks_InsertChunk, 1085
 CF_ChunkSize_t, 1077
CF_Chunk_MAX
 cf_chunk.h, 1077
CF_Chunk_t
 cf_chunk.h, 1076
CF_ChunkIdx_t
 cf_chunk.h, 1076
CF_ChunkList, 577
 chunks, 578
 count, 578
 max_chunks, 578
CF_ChunkList_ComputeGapFn_t
 cf_chunk.h, 1076
CF_ChunkList_ComputeGaps
 cf_chunk.c, 1066
 cf_chunk.h, 1077
CF_ChunkList_GetFirstChunk
 cf_chunk.c, 1067
 cf_chunk.h, 1078
CF_ChunkList_RemoveFromFirst
 cf_chunk.c, 1068
 cf_chunk.h, 1078
CF_ChunkList_t
 cf_chunk.h, 1077
CF_ChunkListAdd
 cf_chunk.c, 1068
 cf_chunk.h, 1079
CF_ChunkListInit
 cf_chunk.c, 1069
 cf_chunk.h, 1080
CF_ChunkListReset
 cf_chunk.c, 1070
 cf_chunk.h, 1080
CF_ChunkOffset_t
 cf_chunk.h, 1077
CF_Chunks_CombineNext
 cf_chunk.c, 1070
 cf_chunk.h, 1081
CF_Chunks_CombinePrevious
 cf_chunk.c, 1071
 cf_chunk.h, 1082
CF_Chunks_EraseChunk
 cf_chunk.c, 1072
 cf_chunk.h, 1082
CF_Chunks_EraseRange
 cf_chunk.c, 1072
 cf_chunk.h, 1083
CF_Chunks_FindInsertPosition
 cf_chunk.c, 1072
 cf_chunk.h, 1083
CF_Chunks_FindSmallestSize
 cf_chunk.c, 1073
 cf_chunk.h, 1084
CF_Chunks_Insert
 cf_chunk.c, 1073
 cf_chunk.h, 1084
CF_Chunks_InsertChunk
 cf_chunk.c, 1074
 cf_chunk.h, 1085
CF_ChunkSize_t
 cf_chunk.h, 1077

CF_ChunkWrapper, 578
 chunks, 579
 cl_node, 579
CF_ChunkWrapper_t
 cf_cfdp_types.h, 1062
cf_clist.c
 CF_CList_InitNode, 1086
 CF_CList_InsertAfter, 1087
 CF_CList_InsertBack, 1087
 CF_CList_InsertFront, 1087
 CF_CList_Pop, 1088
 CF_CList_Remove, 1088
 CF_CList_Traverse, 1089
 CF_CList_Traverse_R, 1090
cf_clist.h
 CF_CLIST_CONT, 1092
 CF_CLIST_EXIT, 1092
 CF_CList_InitNode, 1093
 CF_CList_InsertAfter, 1093
 CF_CList_InsertBack, 1094
 CF_CList_InsertFront, 1094
 CF_CList_Pop, 1094
 CF_CList_Remove, 1095
 CF_CList_Traverse, 1096
 CF_CList_Traverse_R, 1096
 CF_CListFn_t, 1092
 CF_CListNode_t, 1092
 CF_CListTraverse_Status_CONTINUE, 1093
 CF_CListTraverse_Status_EXIT, 1093
 CF_CListTraverse_Status_IS_CONTINUE, 1097
 CF_CListTraverse_Status_t, 1093
 container_of, 1092
CF_CLIST_CONT
 cf_clist.h, 1092
CF_CLIST_EXIT
 cf_clist.h, 1092
CF_CList_InitNode
 cf_clist.c, 1086
 cf_clist.h, 1093
CF_CList_InsertAfter
 cf_clist.c, 1087
 cf_clist.h, 1093
CF_CList_InsertAfter_Ex
 cf_utils.h, 1261
CF_CList_InsertBack
 cf_clist.c, 1087
 cf_clist.h, 1094
CF_CList_InsertBack_Ex
 cf_utils.h, 1261
CF_CList_InsertFront
 cf_clist.c, 1087
 cf_clist.h, 1094
CF_CList_Pop
 cf_clist.c, 1088
 cf_clist.h, 1094
CF_CList_Remove
 cf_clist.c, 1088
 cf_clist.h, 1095
CF_CList_Remove_Ex
 cf_utils.h, 1262
CF_CList_Traverse
 cf_clist.c, 1089
 cf_clist.h, 1096
CF_CList_Traverse_R
 cf_clist.c, 1090
 cf_clist.h, 1096
CF_CListFn_t
 cf_clist.h, 1092
CF_CListNode_t
 cf_clist.h, 1092
 next, 579
 prev, 579
CF_CListTraverse_Status_CONTINUE
 cf_clist.h, 1093
CF_CListTraverse_Status_EXIT
 cf_clist.h, 1093
CF_CListTraverse_Status_IS_CONTINUE
 cf_clist.h, 1097
CF_CListTraverse_Status_t
 cf_clist.h, 1093
cf_cmd.c
 CF_Abandon_TxnCmd, 1099
 CF_AbandonCmd, 1100
 CF_Cancel_TxnCmd, 1101
 CF_CancelCmd, 1102
 CF_DisableDequeueCmd, 1103
 CF_DisableDirPollingCmd, 1104
 CF_DisableEngineCmd, 1104
 CF_DoChanAction, 1105
 CF_DoEnableDisableDequeue, 1106
 CF_DoEnableDisablePolldir, 1106
 CF_DoFreezeThaw, 1107
 CF_DoPurgeQueue, 1107
 CF_DoSuspRes, 1109
 CF_DoSuspRes_Txn, 1110
 CF_EnableDequeueCmd, 1110
 CF_EnableDirPollingCmd, 1111
 CF_EnableEngineCmd, 1111
 CF_FindTransactionBySequenceNumberAllChannels, 1112
 CF_FreezeCmd, 1113
 CF_GetParamCmd, 1114
 CF_SetParamCmd, 1114
 CF_NoopCmd, 1116
 CF_PlaybackDirCmd, 1117
 CF_PurgeHistory, 1117
 CF_PurgeQueueCmd, 1118

CF_PurgeTransaction, 1119
CF_ResetCountersCmd, 1120
CF_ResumeCmd, 1121
CF_SendHkCmd, 1121
CF_SetParamCmd, 1122
CF_SuspendCmd, 1123
CF_ThawCmd, 1123
CF_TsnChanAction, 1124
CF_TxFileCmd, 1125
CF_ValidateChunkSizeCmd, 1126
CF_ValidateMaxOutgoingCmd, 1126
CF_WakeupCmd, 1127
CF_WriteQueueCmd, 1128

cf_cmd.h

CF_Abandon_TxnCmd, 1132
CF_AbandonCmd, 1133
CF_Cancel_TxnCmd, 1134
CF_CancelCmd, 1135
CF_ChanAction_BoolArg_t, 1131
CF_ChanAction_BoolMsgArg_t, 1132
CF_ChanAction_MsgArg_t, 1132
CF_ChanAction_Status_ERROR, 1132
CF_ChanAction_Status_IS_SUCCESS, 1136
CF_ChanAction_Status_SUCCESS, 1132
CF_ChanAction_Status_t, 1132
CF_ChanAction_SuspResArg_t, 1132
CF_ChanActionFn_t, 1132
CF_DisableDequeueCmd, 1136
CF_DisableDirPollingCmd, 1137
CF_DisableEngineCmd, 1137
CF_DoChanAction, 1138
CF_DoEnableDisableDequeue, 1139
CF_DoEnableDisablePolldir, 1139
CF_DoFreezeThaw, 1140
CF_DoPurgeQueue, 1141
CF_DoSuspRes, 1142
CF_DoSuspRes_Txn, 1143
CF_EnableDequeueCmd, 1143
CF_EnableDirPollingCmd, 1144
CF_EnableEngineCmd, 1144
CF_FindTransactionBySequenceNumberAllChannels,
 1145
CF_FreezeCmd, 1146
CF_GetParamCmd, 1147
CF_SetParamCmd, 1147
CF_NoopCmd, 1149
CF_PlaybackDirCmd, 1150
CF_PurgeHistory, 1150
CF_PurgeQueueCmd, 1151
CF_PurgeTransaction, 1152
CF_ResetCountersCmd, 1153
CF_ResumeCmd, 1154
CF_SendHkCmd, 1154
CF_SetParamCmd, 1155

CF_SuspendCmd, 1156
CF_ThawCmd, 1156
CF_TsnChanAction, 1157
CF_TsnChanAction_fn_t, 1132
CF_TxFileCmd, 1158
CF_ValidateChunkSizeCmd, 1159
CF_ValidateMaxOutgoingCmd, 1159
CF_WakeupCmd, 1160
CF_WriteQueueCmd, 1161

CF_CMD_ABANDON_CHAN_ERR_EID
 CFS CFDP Event IDs, 169

CF_CMD_ABANDON_INF_EID
 CFS CFDP Event IDs, 170

CF_CMD_BAD_PARAM_ERR_EID
 CFS CFDP Event IDs, 170

CF_CMD_CANCEL_CHAN_ERR_EID
 CFS CFDP Event IDs, 170

CF_CMD_CANCEL_INF_EID
 CFS CFDP Event IDs, 170

CF_CMD_CHAN_PARAM_ERR_EID
 CFS CFDP Event IDs, 171

CF_CMD_DISABLE_DEQUEUE_ERR_EID
 CFS CFDP Event IDs, 171

CF_CMD_DISABLE_DEQUEUE_INF_EID
 CFS CFDP Event IDs, 171

CF_CMD_DISABLE_ENGINE_INF_EID
 CFS CFDP Event IDs, 171

CF_CMD_DISABLE_POLLDIR_ERR_EID
 CFS CFDP Event IDs, 172

CF_CMD_DISABLE_POLLDIR_INF_EID
 CFS CFDP Event IDs, 172

CF_CMD_ENABLE_DEQUEUE_ERR_EID
 CFS CFDP Event IDs, 172

CF_CMD_ENABLE_DEQUEUE_INF_EID
 CFS CFDP Event IDs, 172

CF_CMD_ENABLE_ENGINE_ERR_EID
 CFS CFDP Event IDs, 173

CF_CMD_ENABLE_ENGINE_INF_EID
 CFS CFDP Event IDs, 173

CF_CMD_ENABLE_POLLDIR_ERR_EID
 CFS CFDP Event IDs, 173

CF_CMD_ENABLE_POLLDIR_INF_EID
 CFS CFDP Event IDs, 173

CF_CMD_ENG_ALREADY_DIS_INF_EID
 CFS CFDP Event IDs, 174

CF_CMD_ENG_ALREADY_ENA_INF_EID
 CFS CFDP Event IDs, 174

CF_CMD_FREEZE_ERR_EID
 CFS CFDP Event IDs, 174

CF_CMD_FREEZE_INF_EID
 CFS CFDP Event IDs, 174

CF_CMD_GETSET1_INF_EID
 CFS CFDP Event IDs, 175

CF_CMD_GETSET2_INF_EID

CFS CFDP Event IDs, [175](#)
CF_CMD_GETSET_CHAN_ERR_EID
 CFS CFDP Event IDs, [175](#)
CF_CMD_GETSET_PARAM_ERR_EID
 CFS CFDP Event IDs, [175](#)
CF_CMD_GETSET_VALIDATE_ERR_EID
 CFS CFDP Event IDs, [176](#)
CF_CMD_LEN_ERR_EID
 CFS CFDP Event IDs, [176](#)
CF_CMD_MID
 CFS CFDP Command Message IDs, [231](#)
CF_CMD_PLAYBACK_DIR_ERR_EID
 CFS CFDP Event IDs, [176](#)
CF_CMD_PLAYBACK_DIR_INF_EID
 CFS CFDP Event IDs, [176](#)
CF_CMD_POLLDIR_INVALID_ERR_EID
 CFS CFDP Event IDs, [177](#)
CF_CMD_PURGE_ARG_ERR_EID
 CFS CFDP Event IDs, [177](#)
CF_CMD_PURGE_QUEUE_ERR_EID
 CFS CFDP Event IDs, [177](#)
CF_CMD_PURGE_QUEUE_INF_EID
 CFS CFDP Event IDs, [177](#)
CF_CMD_RESET_INVALID_ERR_EID
 CFS CFDP Event IDs, [178](#)
CF_CMD_SUSPRES_CHAN_ERR_EID
 CFS CFDP Event IDs, [178](#)
CF_CMD_SUSPRES_INF_EID
 CFS CFDP Event IDs, [178](#)
CF_CMD_SUSPRES_SAME_INF_EID
 CFS CFDP Event IDs, [178](#)
CF_CMD_THAW_ERR_EID
 CFS CFDP Event IDs, [179](#)
CF_CMD_THAW_INF_EID
 CFS CFDP Event IDs, [179](#)
CF_CMD_TRANS_NOT_FOUND_ERR_EID
 CFS CFDP Event IDs, [179](#)
CF_CMD_TSN_CHAN_INVALID_ERR_EID
 CFS CFDP Event IDs, [179](#)
CF_CMD_TX_FILE_ERR_EID
 CFS CFDP Event IDs, [180](#)
CF_CMD_TX_FILE_INF_EID
 CFS CFDP Event IDs, [180](#)
CF_CMD_WHIST_WRITE_ERR_EID
 CFS CFDP Event IDs, [180](#)
CF_CMD_WQ_ARGS_ERR_EID
 CFS CFDP Event IDs, [180](#)
CF_CMD_WQ_CHAN_ERR_EID
 CFS CFDP Event IDs, [181](#)
CF_CMD_WQ_INF_EID
 CFS CFDP Event IDs, [181](#)
CF_CMD_WQ_OPEN_ERR_EID
 CFS CFDP Event IDs, [181](#)
CF_CMD_WQ_WRITEHIST_RX_ERR_EID
 CFS CFDP Event IDs, [181](#)
CF_CMD_WQ_WRITEHIST_TX_ERR_EID
 CFS CFDP Event IDs, [182](#)
CF_CMD_WQ_WRITEQ_PEND_ERR_EID
 CFS CFDP Event IDs, [182](#)
CF_CMD_WQ_WRITEQ_RX_ERR_EID
 CFS CFDP Event IDs, [182](#)
CF_CMD_WQ_WRITEQ_TX_ERR_EID
 CFS CFDP Event IDs, [182](#)
CF_CMDS
 CFS CFDP Command Codes, [195](#)
cf_codec.c
 CF_CFDP_CodecCheckSize, [1165](#)
 CF_CFDP_DecodeAck, [1166](#)
 CF_CFDP_DecodeAllSegments, [1167](#)
 CF_CFDP_DecodeAllTlv, [1167](#)
 CF_CFDP_DecodeCrc, [1168](#)
 CF_CFDP_DecodeEof, [1168](#)
 CF_CFDP_DecodeFileDataHeader, [1169](#)
 CF_CFDP_DecodeFileDirectiveHeader, [1170](#)
 CF_CFDP_DecodeFin, [1170](#)
 CF_CFDP_DecodeHeader, [1171](#)
 CF_CFDP_DecodeLv, [1172](#)
 CF_CFDP_DecodeMd, [1172](#)
 CF_CFDP_DecodeNak, [1173](#)
 CF_CFDP_DecodeSegmentRequest, [1173](#)
 CF_CFDP_DecodeTlv, [1174](#)
 CF_CFDP_DoDecodeChunk, [1174](#)
 CF_CFDP_DoEncodeChunk, [1175](#)
 CF_CFDP_EncodeAck, [1176](#)
 CF_CFDP_EncodeAllSegments, [1176](#)
 CF_CFDP_EncodeAllTlv, [1177](#)
 CF_CFDP_EncodeCrc, [1177](#)
 CF_CFDP_EncodeEof, [1178](#)
 CF_CFDP_EncodeFileDataHeader, [1179](#)
 CF_CFDP_EncodeFileDirectiveHeader, [1179](#)
 CF_CFDP_EncodeFin, [1180](#)
 CF_CFDP_EncodeHeaderFinalSize, [1181](#)
 CF_CFDP_EncodeHeaderWithoutSize, [1181](#)
 CF_CFDP_EncodeLv, [1182](#)
 CF_CFDP_EncodeMd, [1183](#)
 CF_CFDP_EncodeNak, [1183](#)
 CF_CFDP_EncodeSegmentRequest, [1184](#)
 CF_CFDP_EncodeTlv, [1185](#)
 CF_CFDP_GetValueEncodedSize, [1185](#)
 CF_CFDP_PduAck_CC, [1189](#)
 CF_CFDP_PduAck_DIR_CODE, [1189](#)
 CF_CFDP_PduAck_DIR_SUBTYPE_CODE, [1189](#)
 CF_CFDP_PduAck_TRANSACTION_STATUS, [1189](#)
 CF_CFDP_PduEof_FLAGS_CC, [1189](#)
 CF_CFDP_PduFileData_RECORD_CONTINUATION_STATE, [1189](#)
 CF_CFDP_PduFileData_SEGMENT_METADATA_LENGTH, [1189](#)

CF_CFDP_PduFin_FLAGS_CC, 1189
CF_CFDP_PduFin_FLAGS_DELIVERY_CODE,
 1190
CF_CFDP_PduFin_FLAGS_FILE_STATUS, 1190
CF_CFDP_PduHeader_FLAGS_CRC, 1190
CF_CFDP_PduHeader_FLAGS_DIR, 1190
CF_CFDP_PduHeader_FLAGS_LARGEFILE, 1190
CF_CFDP_PduHeader_FLAGS_MODE, 1190
CF_CFDP_PduHeader_FLAGS_TYPE, 1190
CF_CFDP_PduHeader_FLAGS_VERSION, 1190
CF_CFDP_PduHeader_LENGTHS_ENTITY, 1191
CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE, 1191
CF_CFDP_PduMd_CHECKSUM_TYPE, 1191
CF_CFDP_PduMd_CLOSURE_REQUESTED, 1191
CF_Codec_BitField_t, 1165
CF_Codec_Load_uint16, 1186
CF_Codec_Load_uint32, 1186
CF_Codec_Load_uint64, 1186
CF_Codec_Load_uint8, 1186
CF_Codec_Store_uint16, 1186
CF_Codec_Store_uint32, 1186
CF_Codec_Store_uint64, 1186
CF_Codec_Store_uint8, 1187
CF_DecodeIntegerInSize, 1187
CF_EncodeIntegerInSize, 1188
CF_FieldGetVal, 1188
CF_FieldSetVal, 1188
CF_INIT_FIELD, 1165
FGV, 1165
FSV, 1165
cf_codec.h
 CF_CFDP_CodecCheckSize, 1196
 CF_CFDP_CodecGetPosition, 1197
 CF_CFDP_CodecGetRemain, 1198
 CF_CFDP_CodecGetSize, 1198
 CF_CFDP_CodeclsOK, 1198
 CF_CFDP_CodecReset, 1199
 CF_CFDP_CodecSetDone, 1199
 CF_CFDP_DecodeAck, 1199
 CF_CFDP_DecodeAllSegments, 1200
 CF_CFDP_DecodeAllTlv, 1200
 CF_CFDP_DecodeCrc, 1201
 CF_CFDP_DecodeEof, 1201
 CF_CFDP_DecodeFileDataHeader, 1202
 CF_CFDP_DecodeFileDirectiveHeader, 1202
 CF_CFDP_DecodeFin, 1203
 CF_CFDP_DecodeHeader, 1204
 CF_CFDP_DecodeLv, 1205
 CF_CFDP_DecodeMd, 1205
 CF_CFDP_DecodeNak, 1206
 CF_CFDP_DecodeSegmentRequest, 1206
 CF_CFDP_DecodeTlv, 1207
 CF_CFDP_DoDecodeChunk, 1207
 CF_CFDP_DoEncodeChunk, 1208
 CF_CFDP_EncodeAck, 1209
 CF_CFDP_EncodeAllSegments, 1209
 CF_CFDP_EncodeAllTlv, 1210
 CF_CFDP_EncodeCrc, 1210
 CF_CFDP_EncodeEof, 1211
 CF_CFDP_EncodeFileDataHeader, 1212
 CF_CFDP_EncodeFileDirectiveHeader, 1212
 CF_CFDP_EncodeFin, 1213
 CF_CFDP_EncodeHeaderFinalSize, 1214
 CF_CFDP_EncodeHeaderWithoutSize, 1214
 CF_CFDP_EncodeLv, 1215
 CF_CFDP_EncodeMd, 1216
 CF_CFDP_EncodeNak, 1216
 CF_CFDP_EncodeSegmentRequest, 1217
 CF_CFDP_EncodeTlv, 1218
 CF_CFDP_GetValueEncodedSize, 1218
 CF_CODEC_GET_POSITION, 1194
 CF_CODEC_GET_REMAIN, 1194
 CF_CODEC_GET_SIZE, 1194
 CF_CODEC_IS_OK, 1195
 CF_CODEC_SET_DONE, 1195
 CF_CodecState_t, 1196
 CF_DECODE_FIXED_CHUNK, 1195
 CF_DecodeIntegerInSize, 1219
 CF_DecoderState_t, 1196
 CF_ENCODE_FIXED_CHUNK, 1196
 CF_EncodeIntegerInSize, 1219
 CF_EncoderState_t, 1196
 CF_Codec_BitField, 579
 mask, 580
 shift, 580
 CF_Codec_BitField_t
 cf_codec.c, 1165
 CF_CODEC_GET_POSITION
 cf_codec.h, 1194
 CF_CODEC_GET_REMAIN
 cf_codec.h, 1194
 CF_CODEC_GET_SIZE
 cf_codec.h, 1194
 CF_CODEC_IS_OK
 cf_codec.h, 1195
 CF_Codec_Load_uint16
 cf_codec.c, 1186
 CF_Codec_Load_uint32
 cf_codec.c, 1186
 CF_Codec_Load_uint64
 cf_codec.c, 1186
 CF_Codec_Load_uint8
 cf_codec.c, 1186
 CF_CODEC_SET_DONE
 cf_codec.h, 1195
 CF_Codec_Store_uint16
 cf_codec.c, 1186

CF_Codec_Store_uint32
 cf_codec.c, 1186

CF_Codec_Store_uint64
 cf_codec.c, 1186

CF_Codec_Store_uint8
 cf_codec.c, 1187

CF_CodecState, 580
 is_valid, 580
 max_size, 580
 next_offset, 580

CF_CodecState_t
 cf_codec.h, 1196

CF_COMPOUND_KEY
 default_cf_msg.h, 834

CF_config_table
 cf_def_config.c, 1281

CF_CONFIG_TABLE_FILENAME
 CFS CFDP Platform Configuration, 217

CF_CONFIG_TABLE_NAME
 CFS CFDP Platform Configuration, 217

CF_ConfigTable, 581
 chan, 581
 fail_dir, 581
 local_eid, 582
 outgoing_file_chunk_size, 582
 rx_crc_calc_bytes_per_wakeup, 582
 ticks_per_second, 582
 tmp_dir, 582

CF_ConfigTable_t
 default_cf_tblstruct.h, 841

CF_CR_CHANNEL_PIPE_ERR_EID
 CFS CFDP Event IDs, 183

CF_CR_PIPE_ERR_EID
 CFS CFDP Event IDs, 183

CF_Crc, 582
 index, 583
 result, 583
 working, 583

cf_crc.c
 CF_CRC_Digest, 1221
 CF_CRC_Finalize, 1221
 CF_CRC_Start, 1221

cf_crc.h
 CF_CRC_Digest, 1222
 CF_CRC_Finalize, 1223
 CF_CRC_Start, 1223
 CF_Crc_t, 1222

CF_CRC_Digest
 cf_crc.c, 1221
 cf_crc.h, 1222

CF_CRC_Finalize
 cf_crc.c, 1221
 cf_crc.h, 1223

CF_CRC_Start
 cf_crc.c, 1221
 cf_crc.h, 1223

cf_crc.c, 1221
 cf_crc.h, 1223

CF_Crc_t
 cf_crc.h, 1222

CF_DECODE_FIXED_CHUNK
 cf_codec.h, 1195

CF_DecodeIntegerInSize
 cf_codec.c, 1187
 cf_codec.h, 1219

CF_DecoderState, 583
 base, 583
 codec_state, 583

CF_DecoderState_t
 cf_codec.h, 1196

cf_def_config.c
 CF_config_table, 1281

CF_DequeueTransaction
 cf_utils.h, 1262

CF_Direction_NUM
 cf_cfdp_types.h, 1063

CF_Direction_RX
 cf_cfdp_types.h, 1063

CF_Direction_t
 cf_cfdp_types.h, 1063

CF_Direction_TX
 cf_cfdp_types.h, 1063

CF_DISABLE_DEQUEUE_CC
 CFS CFDP Command Codes, 210

CF_DISABLE_DIR_POLLING_CC
 CFS CFDP Command Codes, 212

CF_DISABLE_ENGINE_CC
 CFS CFDP Command Codes, 215

CF_DisableDequeueCmd, 584
 cf_cmd.c, 1103
 cf_cmd.h, 1136
 CommandHeader, 584
 Payload, 584

CF_DisableDequeueCmd_t
 CFS CFDP Command Structures, 227

CF_DisableDirPollingCmd, 584
 cf_cmd.c, 1104
 cf_cmd.h, 1137
 CommandHeader, 585
 Payload, 585

CF_DisableDirPollingCmd_t
 CFS CFDP Command Structures, 227

CF_DisableEngineCmd, 585
 cf_cmd.c, 1104
 cf_cmd.h, 1137
 CommandHeader, 585

CF_DisableEngineCmd_t
 CFS CFDP Command Structures, 227

cf_dispatch.c
 CF_AppPipe, 1224

CF_ProcessGroundCommand, 1225
cf_dispatch.h
 CF_AppPipe, 1226
 CF_ProcessGroundCommand, 1227
CF_DoChanAction
 cf_cmd.c, 1105
 cf_cmd.h, 1138
CF_DoEnableDisableDequeue
 cf_cmd.c, 1106
 cf_cmd.h, 1139
CF_DoEnableDisablePolldir
 cf_cmd.c, 1106
 cf_cmd.h, 1139
CF_DoFreezeThaw
 cf_cmd.c, 1107
 cf_cmd.h, 1140
CF_DoPurgeQueue
 cf_cmd.c, 1107
 cf_cmd.h, 1141
CF_DoSuspRes
 cf_cmd.c, 1109
 cf_cmd.h, 1142
CF_DoSuspRes_Txn
 cf_cmd.c, 1110
 cf_cmd.h, 1143
cf_eds_dispatch.c
 CF_AppPipe, 1229
 CF_TC_DISPATCH_TABLE, 1230
CF_ENABLE_DEQUEUE_CC
 CFS CFDP Command Codes, 209
CF_ENABLE_DIR_POLLING_CC
 CFS CFDP Command Codes, 211
CF_ENABLE_ENGINE_CC
 CFS CFDP Command Codes, 214
CF_EnableDequeueCmd, 585
 cf_cmd.c, 1110
 cf_cmd.h, 1143
 CommandHeader, 586
 Payload, 586
CF_EnableDequeueCmd_t
 CFS CFDP Command Structures, 227
CF_EnableDirPollingCmd, 586
 cf_cmd.c, 1111
 cf_cmd.h, 1144
 CommandHeader, 586
 Payload, 587
CF_EnableDirPollingCmd_t
 CFS CFDP Command Structures, 227
CF_EnableEngineCmd, 587
 cf_cmd.c, 1111
 cf_cmd.h, 1144
 CommandHeader, 587
CF_EnableEngineCmd_t
 CFS CFDP Command Structures, 227
CF_ENCODE_FIXED_CHUNK
 cf_codec.h, 1196
CF_EncodeIntegerInSize
 cf_codec.c, 1188
 cf_codec.h, 1219
CF_EncoderState, 587
 base, 588
 codec_state, 588
CF_EncoderState_t
 cf_codec.h, 1196
CF_Engine, 588
 channels, 588
 chunk_mem, 589
 chunks, 589
 enabled, 589
 histories, 589
 in, 589
 out, 589
 outgoing_counter, 589
 seq_num, 589
 transactions, 589
CF_Engine_t
 cf_cfdp_types.h, 1062
CF_EntityId_t
 default_cf_extern_typedefs.h, 829
CF_EOT_TLM_MID
 CFS CFDP Telemetry Message IDs, 232
CF_EotPacket, 590
 Payload, 590
 TelemetryHeader, 590
CF_EotPacket_Payload, 590
 channel, 591
 crc_result, 591
 direction, 591
 fnames, 591
 fsize, 591
 peer_eid, 591
 seq_num, 592
 src_eid, 592
 state, 592
 txn_stat, 592
CF_EotPacket_Payload_t
 CFS CFDP Telemetry, 223
CF_EotPacket_t
 CFS CFDP Telemetry, 223
CF_ERROR
 cf_app.h, 857
CF_FieldGetVal
 cf_codec.c, 1188
CF_FieldSetVal
 cf_codec.c, 1188
CF_FILENAME_MAX_LEN
 CFS CFDP Platform Configuration, 218
CF_FILENAME_MAX_NAME

CFS CFDP Platform Configuration, 218
CF_FILENAME_MAX_PATH
 default_cf_mission_cfg.h, 833
CF_FileSize_t
 cf_logical_pdu.h, 1233
CF_FindTransactionBySequenceNumber
 cf_utils.c, 1241
 cf_utils.h, 1263
CF_FindTransactionBySequenceNumber_Impl
 cf_utils.c, 1242
 cf_utils.h, 1264
CF_FindTransactionBySequenceNumberAllChannels
 cf_cmd.c, 1112
 cf_cmd.h, 1145
CF_FindUnusedTransaction
 cf_utils.c, 1243
 cf_utils.h, 1264
CF_Flags_Common, 592
 ack_timer_armed, 592
 canceled, 593
 crc_calc, 593
 q_index, 593
 suspended, 593
CF_Flags_Common_t
 cf_cfdp_types.h, 1062
CF_Flags_Rx, 593
 com, 594
 complete, 594
 eof_recv, 594
 fd_nak_sent, 594
 inactivity_fired, 594
 md_recv, 594
 send_ack, 594
 send_fin, 594
 send_nak, 595
CF_Flags_Rx_t
 cf_cfdp_types.h, 1062
CF_Flags_Tx, 595
 cmd_tx, 595
 com, 595
 md_need_send, 595
CF_Flags_Tx_t
 cf_cfdp_types.h, 1062
CF_FreeTransaction
 cf_utils.c, 1243
 cf_utils.h, 1265
CF_FREEZE_CC
 CFS CFDP Command Codes, 200
CF_FreezeCmd, 595
 cf_cmd.c, 1113
 cf_cmd.h, 1146
 CommandHeader, 596
 Payload, 596
CF_FreezeCmd_t
 CFS CFDP Command Structures, 227
CF_GapComputeArgs_t, 596
 nak, 596
 txn, 597
CF_GET_PARAM_CC
 CFS CFDP Command Codes, 207
CF_GetParam_Payload, 597
 chan_num, 597
 key, 597
CF_GetParam_Payload_t
 CFS CFDP Command Structures, 227
CF_GetParamCmd, 597
 cf_cmd.c, 1114
 cf_cmd.h, 1147
 CommandHeader, 598
 Payload, 598
CF_GetParamCmd_t
 CFS CFDP Command Structures, 227
CF_Set_ValueID_ack_limit
 CFS CFDP Command Structures, 229
CF_Set_ValueID_ack_timer_s
 CFS CFDP Command Structures, 229
CF_Set_ValueID_chan_max_outgoing_messages_per_wakeup
 CFS CFDP Command Structures, 229
CF_Set_ValueID_inactivity_timer_s
 CFS CFDP Command Structures, 229
CF_Set_ValueID_chan_max_outgoing_messages_per_wakeup
 CFS CFDP Command Structures, 229
CF_Set_ValueID_local_eid
 CFS CFDP Command Structures, 229
CF_Set_ValueID_MAX
 CFS CFDP Command Structures, 229
CF_Set_ValueID_nak_limit
 CFS CFDP Command Structures, 229
CF_Set_ValueID_nak_timer_s
 CFS CFDP Command Structures, 229
CF_Set_ValueID_outgoing_file_chunk_size
 CFS CFDP Command Structures, 229
CF_Set_ValueID_rx_crc_calc_bytes_per_wakeup
 CFS CFDP Command Structures, 229
CF_Set_ValueID_t
 CFS CFDP Command Structures, 229
CF_Set_ValueID_ticks_per_second
 CFS CFDP Command Structures, 229
CF_SetParamCmd
 cf_cmd.c, 1114
 cf_cmd.h, 1147
CF_History, 598
 cl_node, 599
 dir, 599
 fnames, 599
 peer_eid, 599
 seq_num, 599
 src_eid, 599
 txn_stat, 600
CF_History_t

cf_cfdp_types.h, 1062
CF_HK_TLM_MID
 CFS CFDP Telemetry Message IDs, 232
CF_HkChannel_Data, 600
 counters, 600
 frozen, 601
 playback_counter, 601
 poll_counter, 601
 q_size, 601
 spare, 601
CF_HkChannel_Data_t
 CFS CFDP Telemetry, 223
CF_HkCmdCounters, 601
 cmd, 602
 err, 602
CF_HkCmdCounters_t
 CFS CFDP Telemetry, 223
CF_HkCounters, 602
 fault, 602
 recv, 603
 sent, 603
CF_HkCounters_t
 CFS CFDP Telemetry, 223
CF_HkFault, 603
 ack_limit, 604
 crc_mismatch, 604
 directory_read, 604
 file_open, 604
 file_read, 604
 file_rename, 604
 file_seek, 604
 file_size_mismatch, 605
 file_write, 605
 inactivity_timer, 605
 nak_limit, 605
 spare, 605
CF_HkFault_t
 CFS CFDP Telemetry, 223
CF_HkPacket, 605
 Payload, 606
 TelemetryHeader, 606
CF_HkPacket_Payload, 606
 channel_hk, 606
 counters, 607
 spare, 607
CF_HkPacket_Payload_t
 CFS CFDP Telemetry, 223
CF_HkPacket_t
 CFS CFDP Telemetry, 223
CF_HkRecv, 607
 dropped, 608
 error, 608
 file_data_bytes, 608
 nak_segment_requests, 608
 pdu, 608
 spurious, 608
CF_HkRecv_t
 CFS CFDP Telemetry, 223
CF_HkSent, 608
 file_data_bytes, 609
 nak_segment_requests, 609
 pdu, 609
CF_HkSent_t
 CFS CFDP Telemetry, 223
CF_INIT_CRC_ALIGN_ERR_EID
 CFS CFDP Event IDs, 183
CF_INIT_FIELD
 cf_codec.c, 1165
CF_INIT_INF_EID
 CFS CFDP Event IDs, 183
CF_INIT_MSG_RECV_ERR_EID
 CFS CFDP Event IDs, 184
CF_INIT_OUTGOING_SIZE_ERR_EID
 CFS CFDP Event IDs, 184
CF_INIT_SEM_ERR_EID
 CFS CFDP Event IDs, 184
CF_INIT_SUB_ERR_EID
 CFS CFDP Event IDs, 184
CF_INIT_TBL_CHECK_GA_ERR_EID
 CFS CFDP Event IDs, 185
CF_INIT_TBL_CHECK_MAN_ERR_EID
 CFS CFDP Event IDs, 185
CF_INIT_TBL_CHECK_REL_ERR_EID
 CFS CFDP Event IDs, 185
CF_INIT_TBL_GETADDR_ERR_EID
 CFS CFDP Event IDs, 185
CF_INIT_TBL_LOAD_ERR_EID
 CFS CFDP Event IDs, 186
CF_INIT_TBL_MANAGE_ERR_EID
 CFS CFDP Event IDs, 186
CF_INIT_TBL_REG_ERR_EID
 CFS CFDP Event IDs, 186
CF_INIT_TPS_ERR_EID
 CFS CFDP Event IDs, 186
CF_Input, 609
 decode, 610
 msg, 610
 rx_pdudata, 610
CF_Input_t
 cf_cfdp_types.h, 1062
CF_InsertSortPrio
 cf_utils.c, 1244
 cf_utils.h, 1265
CF_Logical_IntHeader, 610
 ack, 611
 eof, 611
 fd, 611
 fin, 611

md, 611
 nak, 611
CF_Logical_IntHeader_t
cf_logical_pdu.h, 1233
CF_Logical_Lv_t, 611
 data_ptr, 612
 length, 612
CF_Logical_Lv_t
cf_logical_pdu.h, 1233
cf_logical_pdu.h
 CF_FileSize_t, 1233
 CF_Logical_IntHeader_t, 1233
 CF_Logical_Lv_t, 1233
 CF_Logical_PduAck_t, 1233
 CF_Logical_PduBuffer_t, 1233
 CF_Logical_PduEof_t, 1233
 CF_Logical_PduFileDataHeader_t, 1233
 CF_Logical_PduFileDirectiveHeader_t, 1233
 CF_Logical_PduFin_t, 1234
 CF_Logical_PduHeader_t, 1234
 CF_Logical_PduMd_t, 1234
 CF_Logical_PduNak_t, 1234
 CF_Logical_SegmentList_t, 1234
 CF_Logical_SegmentRequest_t, 1234
 CF_Logical_Tlv_t, 1234
 CF_Logical_TlvData_t, 1234
 CF_Logical_TlvList_t, 1234
 CF_PDU_MAX_SEGMENTS, 1232
 CF_PDU_MAX_TLV, 1233
CF_Logical_PduAck, 612
 ack_directive_code, 613
 ack_subtype_code, 613
 cc, 613
 txn_status, 613
CF_Logical_PduAck_t
cf_logical_pdu.h, 1233
CF_Logical_PduBuffer, 613
 content_crc, 614
 fdirective, 614
 int_header, 614
 pdec, 614
 pdu_header, 614
 penc, 614
CF_Logical_PduBuffer_t
cf_logical_pdu.h, 1233
CF_Logical_PduEof, 615
 cc, 615
 crc, 615
 size, 615
 tlv_list, 615
CF_Logical_PduEof_t
cf_logical_pdu.h, 1233
CF_Logical_PduFileDataHeader, 615
 continuation_state, 616
 data_len, 616
 data_ptr, 616
 offset, 616
 segment_list, 616
CF_Logical_PduFileDataHeader_t
cf_logical_pdu.h, 1233
CF_Logical_PduFileDirectiveHeader, 617
 directive_code, 617
CF_Logical_PduFileDirectiveHeader_t
cf_logical_pdu.h, 1233
CF_Logical_PduFin, 617
 cc, 617
 delivery_code, 618
 file_status, 618
 tlv_list, 618
CF_Logical_PduFin_t
cf_logical_pdu.h, 1234
CF_Logical_PduHeader, 618
 crc_flag, 619
 data_encoded_length, 619
 destination_eid, 619
 direction, 619
 eid_length, 619
 header_encoded_length, 620
 large_flag, 620
 pdu_type, 620
 segment_meta_flag, 620
 sequence_num, 620
 source_eid, 620
 txm_mode, 620
 txn_seq_length, 620
 version, 621
CF_Logical_PduHeader_t
cf_logical_pdu.h, 1234
CF_Logical_PduMd, 621
 checksum_type, 621
 close_req, 621
 dest_filename, 621
 size, 622
 source_filename, 622
CF_Logical_PduMd_t
cf_logical_pdu.h, 1234
CF_Logical_PduNak, 622
 scope_end, 622
 scope_start, 622
 segment_list, 622
CF_Logical_PduNak_t
cf_logical_pdu.h, 1234
CF_Logical_SegmentList, 623
 num_segments, 623
 segments, 623
CF_Logical_SegmentList_t
cf_logical_pdu.h, 1234
CF_Logical_SegmentRequest, 623

offset_end, 624
offset_start, 624
CF_Logical_SegmentRequest_t
 cf_logical_pdu.h, 1234
CF_Logical_Tlv, 624
 data, 624
 length, 624
 type, 625
CF_Logical_Tlv_t
 cf_logical_pdu.h, 1234
CF_Logical_TlvData, 625
 data_ptr, 625
 eid, 625
CF_Logical_TlvData_t
 cf_logical_pdu.h, 1234
CF_Logical_TlvList, 626
 num_tlv, 626
 tlv, 626
CF_Logical_TlvList_t
 cf_logical_pdu.h, 1234
CF_MAJOR_VERSION
 CFS CFDP Version, 194
CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PERCHAN
 CFS CFDP Platform Configuration, 218
CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN
 CFS CFDP Platform Configuration, 218
CF_MAX_PDU_SIZE
 CFS CFDP Platform Configuration, 218
CF_MAX_POLLING_DIR_PER_CHAN
 CFS CFDP Platform Configuration, 219
CF_MAX_SIMULTANEOUS_RX
 CFS CFDP Platform Configuration, 219
CF_MID_ERR_EID
 CFS CFDP Event IDs, 187
CF_MINOR_VERSION
 CFS CFDP Version, 194
CF_MISSION_REV
 default_cf_platform_cfg.h, 839
CF_MoveTransaction
 cf_utils.h, 1266
CF_NAK_MAX_SEGMENTS
 CFS CFDP Platform Configuration, 219
CF_NOOP_CC
 CFS CFDP Command Codes, 196
CF_NOOP_INF_EID
 CFS CFDP Event IDs, 187
CF_NoopCmd, 626
 cf_cmd.c, 1116
 cf_cmd.h, 1149
 CommandHeader, 626
CF_NoopCmd_t
 CFS CFDP Command Structures, 227
CF_NUM_CHANNELS
 CFS CFDP Platform Configuration, 219
 cf_cfdp_types.h, 1061
CF_NUM_COMMANDS
 CFS CFDP Command Codes, 215
CF_NUM_HISTORIES
 cf_cfdp_types.h, 1061
CF_NUM_HISTORIES_PER_CHANNEL
 CFS CFDP Platform Configuration, 220
CF_NUM_TRANSACTIONS
 cf_cfdp_types.h, 1061
CF_NUM_TRANSACTIONS_PER_CHANNEL
 cf_cfdp_types.h, 1061
CF_NUM_TRANSACTIONS_PER_PLAYBACK
 CFS CFDP Platform Configuration, 220
CF_Output, 627
 encode, 627
 msg, 627
 tx_pdudata, 627
CF_Output_t
 cf_cfdp_types.h, 1062
CF_PDU_ACK_SHORT_ERR_EID
 CFS CFDP Event IDs, 187
 CFS CFDP Platform Configuration, 220
CF_PDU_EOF_SHORT_ERR_EID
 CFS CFDP Event IDs, 187
 CFS CFDP Platform Configuration, 220
CF_PDU_FD_SHORT_ERR_EID
 CFS CFDP Event IDs, 188
 CFS CFDP Platform Configuration, 220
CF_PDU_FD_UNSUPPORTED_ERR_EID
 CFS CFDP Event IDs, 188
 CFS CFDP Platform Configuration, 220
CF_PDU_FIN_SHORT_ERR_EID
 CFS CFDP Event IDs, 188
 CFS CFDP Platform Configuration, 220
CF_PDU_INVALID_DST_LEN_ERR_EID
 CFS CFDP Event IDs, 188
 CFS CFDP Platform Configuration, 220
CF_PDU_INVALID_SRC_LEN_ERR_EID
 CFS CFDP Event IDs, 189
 CFS CFDP Platform Configuration, 220
CF_PDU_LARGE_FILE_ERR_EID
 CFS CFDP Event IDs, 189
 CFS CFDP Platform Configuration, 220
CF_PDU_MAX_SEGMENTS
 cf_logical_pdu.h, 1232
CF_PDU_MAX_TLV
 cf_logical_pdu.h, 1233
CF_PDU_MD_RECVD_INF_EID
 CFS CFDP Event IDs, 189
 CFS CFDP Platform Configuration, 220
CF_PDU_MD_SHORT_ERR_EID
 CFS CFDP Event IDs, 189
 CFS CFDP Platform Configuration, 220
CF_PDU_METADATA_ERROR
 cf_app.h, 857
CF_PDU_NAK_SHORT_ERR_EID
 CFS CFDP Event IDs, 190
 CFS CFDP Platform Configuration, 220
CF_PDU_SHORT_HEADER_ERR_EID
 CFS CFDP Event IDs, 190
 CFS CFDP Platform Configuration, 220
CF_PDU_TRUNCATION_ERR_EID
 CFS CFDP Event IDs, 190
 CFS CFDP Platform Configuration, 220

CF_PduCmdMsg, [627](#)
 hdr, [628](#)
 ph, [628](#)

CF_PduCmdMsg_t
 cf_cfdp_sbintf.h, [1054](#)

CF_PduTlmMsg, [628](#)
 hdr, [629](#)
 ph, [629](#)

CF_PduTlmMsg_t
 cf_cfdp_sbintf.h, [1054](#)

CF_PERF_ID_APPMAIN
 CFS CFDP Mission Configuration, [192](#)

CF_PERF_ID_CREAT
 CFS CFDP Mission Configuration, [192](#)

CF_PERF_ID_CYCLE_ENG
 CFS CFDP Mission Configuration, [192](#)

CF_PERF_ID_DIRREAD
 CFS CFDP Mission Configuration, [192](#)

CF_PERF_ID_FCLOSE
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_FOPEN
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_FREAD
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_FSEEK
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_FWRITE
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_PDURCVD
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_PDUSENT
 CFS CFDP Mission Configuration, [193](#)

CF_PERF_ID_RENAME
 CFS CFDP Mission Configuration, [193](#)

CF_PIPE_DEPTH
 CFS CFDP Platform Configuration, [220](#)

CF_PIPE_NAME
 cf_app.h, [857](#)

CF_Playback, [629](#)
 busy, [629](#)
 cfdp_class, [629](#)
 counted, [630](#)
 dest_id, [630](#)
 dir_id, [630](#)
 diopen, [630](#)
 fnames, [630](#)
 keep, [630](#)
 num_ts, [630](#)
 priority, [630](#)

CF_PLAYBACK_DIR_CC
 CFS CFDP Command Codes, [199](#)

CF_Playback_t
 cf_cfdp_types.h, [1062](#)

CF_PlaybackDirCmd, [631](#)
 cf_cmd.c, [1117](#)
 cf_cmd.h, [1150](#)
 CommandHeader, [631](#)
 Payload, [631](#)

CF_PlaybackDirCmd_t
 CFS CFDP Command Structures, [227](#)

CF_Poll, [631](#)
 interval_timer, [631](#)
 pb, [632](#)
 timer_set, [632](#)

CF_Poll_t
 cf_cfdp_types.h, [1062](#)

CF_PollDir, [632](#)
 cfdp_class, [632](#)
 dest_eid, [632](#)
 dst_dir, [633](#)
 enabled, [633](#)
 interval_sec, [633](#)
 priority, [633](#)
 src_dir, [633](#)

CF_PollDir_t
 default_cf_tbldefs.h, [840](#)

CF_PrioSearch
 cf_utils.c, [1245](#)
 cf_utils.h, [1267](#)

CF_ProcessGroundCommand
 cf_dispatch.c, [1225](#)
 cf_dispatch.h, [1227](#)

CF_PURGE_QUEUE_CC
 CFS CFDP Command Codes, [213](#)

CF_PurgeHistory
 cf_cmd.c, [1117](#)
 cf_cmd.h, [1150](#)

CF_PurgeQueueCmd, [633](#)
 cf_cmd.c, [1118](#)
 cf_cmd.h, [1151](#)
 CommandHeader, [634](#)
 Payload, [634](#)

CF_PurgeQueueCmd_t
 CFS CFDP Command Structures, [227](#)

CF_PurgeTransaction
 cf_cmd.c, [1119](#)
 cf_cmd.h, [1152](#)

CF_Queue_active
 CFS CFDP Command Structures, [229](#)

CF_Queue_all
 CFS CFDP Command Structures, [229](#)

CF_Queue_history
 CFS CFDP Command Structures, [229](#)

CF_Queue_pend
 CFS CFDP Command Structures, [229](#)

CF_Queue_t
 CFS CFDP Command Structures, [229](#)

CF_QueueIdx_FREE

default_cf_extern_typedefs.h, 831
CF_QueueIdx_HIST
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_HIST_FREE
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_NUM
 default_cf_extern_typedefs.h, 831
CF_QueueIdx_PEND
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_RX
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_t
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_TXA
 default_cf_extern_typedefs.h, 830
CF_QueueIdx_TXW
 default_cf_extern_typedefs.h, 830
CF_R2_CRC_CHUNK_SIZE
 CFS CFDP Platform Configuration, 221
CF_RCVMSG_TIMEOUT
 CFS CFDP Platform Configuration, 221
CF_REC_PDU_BAD_EOF_ERROR
 cf_app.h, 857
CF_REC_PDU_FSIZE_MISMATCH_ERROR
 cf_app.h, 857
CF_Reset_all
 CFS CFDP Command Structures, 230
CF_RESET_CC
 CFS CFDP Command Codes, 197
CF_Reset_command
 CFS CFDP Command Structures, 230
CF_Reset_down
 CFS CFDP Command Structures, 230
CF_Reset_fault
 CFS CFDP Command Structures, 230
CF_RESET_FREED_XACT_DBG_EID
 CFS CFDP Event IDs, 190
CF_RESET_INF_EID
 CFS CFDP Event IDs, 191
CF_Reset_t
 CFS CFDP Command Structures, 229
CF_Reset_up
 CFS CFDP Command Structures, 230
CF_ResetCountersCmd, 634
 cf_cmd.c, 1120
 cf_cmd.h, 1153
 CommandHeader, 634
 Payload, 634
CF_ResetCountersCmd_t
 CFS CFDP Command Structures, 228
CF_ResetHistory
 cf_utils.c, 1245
 cf_utils.h, 1267
CF_RESUME_CC
 CFS CFDP Command Codes, 203
CF_ResumeCmd, 635
 cf_cmd.c, 1121
 cf_cmd.h, 1154
 CommandHeader, 635
 Payload, 635
CF_ResumeCmd_t
 CFS CFDP Command Structures, 228
CF_REVISION
 CFS CFDP Version, 194
CF_RxS2_Data, 635
 acknak_count, 636
 dc, 636
 eof_cc, 636
 eof_crc, 636
 eof_size, 636
 fs, 636
 rx_crc_calc_bytes, 636
CF_RxS2_Data_t
 cf_cfdp_types.h, 1063
CF_RxState_Data, 636
 cached_pos, 637
 r2, 637
 sub_state, 637
CF_RxState_Data_t
 cf_cfdp_types.h, 1063
CF_RxSubState_EOF
 cf_cfdp_types.h, 1064
CF_RxSubState_FILEDATA
 cf_cfdp_types.h, 1064
CF_RxSubState_NUM_STATES
 cf_cfdp_types.h, 1064
CF_RxSubState_t
 cf_cfdp_types.h, 1063
CF_RxSubState_WAIT_FOR_FIN_ACK
 cf_cfdp_types.h, 1064
CF_SEND_HK_MID
 CFS CFDP Command Message IDs, 231
CF_SEND_PDU_ERROR
 cf_app.h, 857
CF_SEND_PDU_NO_BUF_AVAIL_ERROR
 cf_app.h, 857
CF_SendHkCmd, 637
 cf_cmd.c, 1121
 cf_cmd.h, 1154
 CommandHeader, 637
CF_SendHkCmd_t
 CFS CFDP Command Structures, 228
CF_SET_PARAM_CC
 CFS CFDP Command Codes, 206
CF_SetParam_Payload, 638
 chan_num, 638
 key, 638
 spare, 638

value, [638](#)
CF_SetParam_Payload_t
 CFS CFDP Command Structures, [228](#)
CF_SetParamCmd, [639](#)
 cf_cmd.c, [1122](#)
 cf_cmd.h, [1155](#)
 CommandHeader, [639](#)
 Payload, [639](#)
CF_SetParamCmd_t
 CFS CFDP Command Structures, [228](#)
CF_SHORT_PDU_ERROR
 cf_app.h, [857](#)
CF_STARTUP_SEM_MAX_RETRIES
 CFS CFDP Platform Configuration, [221](#)
CF_STARTUP_SEM_TASK_DELAY
 CFS CFDP Platform Configuration, [221](#)
CF_StateData, [639](#)
 receive, [640](#)
 send, [640](#)
CF_StateData_t
 cf_cfdp_types.h, [1063](#)
CF_StateFlags, [640](#)
 com, [640](#)
 rx, [641](#)
 tx, [641](#)
CF_StateFlags_t
 cf_cfdp_types.h, [1063](#)
CF_SUSPEND_CC
 CFS CFDP Command Codes, [202](#)
CF_SuspendCmd, [641](#)
 cf_cmd.c, [1123](#)
 cf_cmd.h, [1156](#)
 CommandHeader, [641](#)
 Payload, [641](#)
CF_SuspendCmd_t
 CFS CFDP Command Structures, [228](#)
CF_TableInit
 cf_app.c, [853](#)
 cf_app.h, [860](#)
CF_TC_DISPATCH_TABLE
 cf_eds_dispatch.c, [1230](#)
CF_THAW_CC
 CFS CFDP Command Codes, [201](#)
CF_ThawCmd, [642](#)
 cf_cmd.c, [1123](#)
 cf_cmd.h, [1156](#)
 CommandHeader, [642](#)
 Payload, [642](#)
CF_ThawCmd_t
 CFS CFDP Command Structures, [228](#)
CF_TickType_NUM_TYPES
 cf_cfdp_types.h, [1064](#)
CF_TickType_RX
 cf_cfdp_types.h, [1064](#)
CF_TickType_t
 cf_cfdp_types.h, [1064](#)
CF_TickType_TXW_NAK
 cf_cfdp_types.h, [1064](#)
CF_TickType_TXW_NORM
 cf_cfdp_types.h, [1064](#)
CF_Timer, [642](#)
 tick, [643](#)
cf_timer.c
 CF_Timer_Expired, [1235](#)
 CF_Timer_InitRelSec, [1236](#)
 CF_Timer_Sec2Ticks, [1236](#)
 CF_Timer_Tick, [1237](#)
cf_timer.h
 CF_Timer_Expired, [1238](#)
 CF_Timer_InitRelSec, [1238](#)
 CF_Timer_Sec2Ticks, [1239](#)
 CF_Timer_Seconds_t, [1238](#)
 CF_Timer_t, [1238](#)
 CF_Timer_Tick, [1239](#)
 CF_Timer_Ticks_t, [1238](#)
CF_Timer_Expired
 cf_timer.c, [1235](#)
 cf_timer.h, [1238](#)
CF_Timer_InitRelSec
 cf_timer.c, [1236](#)
 cf_timer.h, [1238](#)
CF_Timer_Sec2Ticks
 cf_timer.c, [1236](#)
 cf_timer.h, [1239](#)
CF_Timer_Seconds_t
 cf_timer.h, [1238](#)
CF_Timer_t
 cf_timer.h, [1238](#)
CF_Timer_Tick
 cf_timer.c, [1237](#)
 cf_timer.h, [1239](#)
CF_Timer_Ticks_t
 cf_timer.h, [1238](#)
CF_TOTAL_CHUNKS
 default_cf_platform_cfg.h, [839](#)
CF_Transaction, [643](#)
 ack_timer, [644](#)
 chan_num, [644](#)
 chunks, [644](#)
 cl_node, [644](#)
 crc, [644](#)
 fd, [644](#)
 flags, [645](#)
 foofs, [645](#)
 fsize, [645](#)
 history, [645](#)
 inactivity_timer, [646](#)
 keep, [646](#)

pb, 646
priority, 646
state, 646
state_data, 646
CF_Transaction_Payload, 647
chan, 647
eid, 647
spare, 647
ts, 647
CF_Transaction_Payload_t
 CFS CFDP Command Structures, 228
CF_Transaction_t
 cf_cfdp_types.h, 1063
CF_TransactionSeq_t
 default_cf_extern_typedefs.h, 829
CF_Traverse_PriorityArg, 648
 priority, 648
 txn, 648
CF_Traverse_PriorityArg_t
 cf_utils.h, 1260
CF_Traverse_TransSeqArg, 648
 src_eid, 649
 transaction_sequence_number, 649
 txn, 649
CF_Traverse_TransSeqArg_t
 cf_utils.h, 1260
CF_Traverse_WriteHistoryFileArg, 649
 counter, 649
 error, 649
 fd, 650
 filter_dir, 650
CF_Traverse_WriteHistoryFileArg_t
 cf_utils.h, 1260
CF_Traverse_WriteHistoryQueueEntryToFile
 cf_utils.c, 1246
 cf_utils.h, 1268
CF_Traverse_WriteTxnFileArg, 650
 counter, 650
 error, 650
 fd, 650
CF_Traverse_WriteTxnFileArg_t
 cf_utils.h, 1260
CF_Traverse_WriteTxnQueueEntryToFile
 cf_utils.c, 1247
 cf_utils.h, 1269
CF_TraverseAll_Arg, 651
 context, 651
 counter, 651
 fn, 651
CF_TraverseAll_Arg_t
 cf_utils.h, 1261
CF_TraverseAllTransactions
 cf_utils.c, 1248
 cf_utils.h, 1269
CF_TraverseAllTransactions_All_Channels
 cf_utils.c, 1248
 cf_utils.h, 1270
CF_TraverseAllTransactions_fn_t
 cf_utils.h, 1261
CF_TraverseAllTransactions_Impl
 cf_utils.c, 1249
 cf_utils.h, 1271
CF_TsnChanAction
 cf_cmd.c, 1124
 cf_cmd.h, 1157
CF_TsnChanAction_fn_t
 cf_cmd.h, 1132
CF_TX_FILE_CC
 CFS CFDP Command Codes, 198
CF_TxFile_Payload, 651
 cfdp_class, 652
 chan_num, 652
 dest_id, 652
 dst_filename, 652
 keep, 652
 priority, 653
 src_filename, 653
CF_TxFile_Payload_t
 CFS CFDP Command Structures, 228
CF_TxFileCmd, 653
 cf_cmd.c, 1125
 cf_cmd.h, 1158
 CommandHeader, 653
 Payload, 653
CF_TxFileCmd_t
 CFS CFDP Command Structures, 228
CF_TxnFilenames, 654
 dst_filename, 654
 src_filename, 654
CF_TxnFilenames_t
 default_cf_extern_typedefs.h, 830
CF_TxnState_DROP
 cf_cfdp_types.h, 1064
CF_TxnState_IDLE
 cf_cfdp_types.h, 1064
CF_TxnState_INVALID
 cf_cfdp_types.h, 1064
CF_TxnState_R1
 cf_cfdp_types.h, 1064
CF_TxnState_R2
 cf_cfdp_types.h, 1064
CF_TxnState_S1
 cf_cfdp_types.h, 1064
CF_TxnState_S2
 cf_cfdp_types.h, 1064
CF_TxnState_t
 cf_cfdp_types.h, 1064
CF_TxnStatus_ACK_LIMIT_NO_EOF

cf_cfdp_types.h, 1065
CF_TxnStatus_ACK_LIMIT_NO_FIN
 cf_cfdp_types.h, 1065
CF_TxnStatus_CANCEL_REQUEST_RECEIVED
 cf_cfdp_types.h, 1065
CF_TxnStatus_CHECK_LIMIT_REACHED
 cf_cfdp_types.h, 1065
CF_TxnStatus_EARLY_FIN
 cf_cfdp_types.h, 1065
CF_TxnStatus_FILE_CHECKSUM_FAILURE
 cf_cfdp_types.h, 1065
CF_TxnStatus_FILE_SIZE_ERROR
 cf_cfdp_types.h, 1065
CF_TxnStatus_FILESTORE_REJECTION
 cf_cfdp_types.h, 1065
CF_TxnStatus_From_ConditionCode
 cf_utils.c, 1250
 cf_utils.h, 1271
CF_TxnStatus_INACTIVITY_DETECTED
 cf_cfdp_types.h, 1065
CF_TxnStatus_INVALID_FILE_STRUCTURE
 cf_cfdp_types.h, 1065
CF_TxnStatus_INVALID_TRANSMISSION_MODE
 cf_cfdp_types.h, 1065
CF_TxnStatus_IsError
 cf_utils.c, 1250
 cf_utils.h, 1272
CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED
 cf_cfdp_types.h, 1065
CF_TxnStatus_MAX
 cf_cfdp_types.h, 1065
CF_TxnStatus_NAK_LIMIT_REACHED
 cf_cfdp_types.h, 1065
CF_TxnStatus_NAK_RESPONSE_ERROR
 cf_cfdp_types.h, 1065
CF_TxnStatus_NO_ERROR
 cf_cfdp_types.h, 1065
CF_TxnStatus_POS_ACK_LIMIT_REACHED
 cf_cfdp_types.h, 1065
CF_TxnStatus_PROTOCOL_ERROR
 cf_cfdp_types.h, 1065
CF_TxnStatus_SEND_EOF_FAILURE
 cf_cfdp_types.h, 1065
CF_TxnStatus_SUSPEND_REQUEST_RECEIVED
 cf_cfdp_types.h, 1065
CF_TxnStatus_t
 cf_cfdp_types.h, 1064
CF_TxnStatus_To_ConditionCode
 cf_utils.c, 1251
 cf_utils.h, 1272
CF_TxnStatus_UNDEFINED
 cf_cfdp_types.h, 1064
CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE
 cf_cfdp_types.h, 1065

CF_TxS2_Data, 654
 acknak_count, 654
 fin_cc, 655
CF_TxS2_Data_t
 cf_cfdp_types.h, 1063
CF_TxState_Data, 655
 cached_pos, 655
 s2, 655
 sub_state, 655
CF_TxState_Data_t
 cf_cfdp_types.h, 1063
CF_TxSubState_EOF
 cf_cfdp_types.h, 1065
CF_TxSubState_FILEDATA
 cf_cfdp_types.h, 1065
CF_TxSubState_METADATA
 cf_cfdp_types.h, 1065
CF_TxSubState_NUM_STATES
 cf_cfdp_types.h, 1065
CF_TxSubState_SEND_FIN_ACK
 cf_cfdp_types.h, 1065
CF_TxSubState_t
 cf_cfdp_types.h, 1065
CF_TxSubState_WAIT_FOR_EOF_ACK
 cf_cfdp_types.h, 1065
CF_TxSubState_WAIT_FOR_FIN
 cf_cfdp_types.h, 1065
CF_Type_all
 CFS CFDP Command Structures, 230
CF_Type_down
 CFS CFDP Command Structures, 230
CF_Type_t
 CFS CFDP Command Structures, 230
CF_Type_up
 CFS CFDP Command Structures, 230
CF_UnionArgs_Payload, 656
 byte, 656
 dword, 656
 hword, 656
CF_UnionArgs_Payload_t
 CFS CFDP Command Structures, 228
cf_utils.c
 CF_FindTransactionBySequenceNumber, 1241
 CF_FindTransactionBySequenceNumber_Impl, 1242
 CF_FindUnusedTransaction, 1243
 CF_FreeTransaction, 1243
 CF_InsertSortPrio, 1244
 CF_PrioSearch, 1245
 CF_ResetHistory, 1245
 CF_Traverse_WriteHistoryQueueEntryToFile, 1246
 CF_Traverse_WriteTxnQueueEntryToFile, 1247
 CF_TraverseAllTransactions, 1248
 CF_TraverseAllTransactions_All_Channels, 1248
 CF_TraverseAllTransactions_Impl, 1249

CF_TxnStatus_From_ConditionCode, 1250
CF_TxnStatus_IsError, 1250
CF_TxnStatus_To_ConditionCode, 1251
CF_WrappedClose, 1251
CF_WrappedLseek, 1252
CF_WrappedOpenCreate, 1253
CF_WrappedRead, 1254
CF_WrappedWrite, 1255
CF_WriteHistoryEntryToFile, 1256
CF_WriteHistoryQueueDataToFile, 1257
CF_WriteTxnQueueDataToFile, 1258
cf_utils.h
 CF_CList_InsertAfter_Ex, 1261
 CF_CList_InsertBack_Ex, 1261
 CF_CList_Remove_Ex, 1262
 CF_DequeueTransaction, 1262
 CF_FindTransactionBySequenceNumber, 1263
 CF_FindTransactionBySequenceNumber_Impl, 1264
 CF_FindUnusedTransaction, 1264
 CF_FreeTransaction, 1265
 CF_InsertSortPrio, 1265
 CF_MoveTransaction, 1266
 CF_PrioSearch, 1267
 CF_ResetHistory, 1267
 CF_Traverse_PriorityArg_t, 1260
 CF_Traverse_TransSeqArg_t, 1260
 CF_Traverse_WriteHistoryFileArg_t, 1260
 CF_Traverse_WriteHistoryQueueEntryToFile, 1268
 CF_Traverse_WriteTxnFileArg_t, 1260
 CF_Traverse_WriteTxnQueueEntryToFile, 1269
 CF_TraverseAll_Arg_t, 1261
 CF_TraverseAllTransactions, 1269
 CF_TraverseAllTransactions_All_Channels, 1270
 CF_TraverseAllTransactions_fn_t, 1261
 CF_TraverseAllTransactions_Impl, 1271
 CF_TxnStatus_From_ConditionCode, 1271
 CF_TxnStatus_IsError, 1272
 CF_TxnStatus_To_ConditionCode, 1272
 CF_WrappedClose, 1273
 CF_WrappedLseek, 1274
 CF_WrappedOpenCreate, 1275
 CF_WrappedRead, 1276
 CF_WrappedWrite, 1277
 CF_WriteHistoryEntryToFile, 1278
 CF_WriteHistoryQueueDataToFile, 1279
 CF_WriteTxnQueueDataToFile, 1280
CF_ValidateChunkSizeCmd
 cf_cmd.c, 1126
 cf_cmd.h, 1159
CF_ValidateConfigTable
 cf_app.c, 854
 cf_app.h, 861
CF_ValidateMaxOutgoingCmd
 cf_cmd.c, 1126
 cf_cmd.h, 1159
 cf_utils.c, 1256
 cf_utils.h, 1273
CF_WAKE_UP_MID
 CFS CFDP Command Message IDs, 231
CF_WakeupCmd, 656
 cf_cmd.c, 1127
 cf_cmd.h, 1160
 CommandHeader, 657
CF_WakeupCmd_t
 CFS CFDP Command Structures, 228
CF_WrappedClose
 cf_utils.c, 1251
 cf_utils.h, 1273
CF_WrappedLseek
 cf_utils.c, 1252
 cf_utils.h, 1274
CF_WrappedOpenCreate
 cf_utils.c, 1253
 cf_utils.h, 1275
CF_WrappedRead
 cf_utils.c, 1254
 cf_utils.h, 1276
CF_WrappedWrite
 cf_utils.c, 1255
 cf_utils.h, 1277
CF_WRITE_QUEUE_CC
 CFS CFDP Command Codes, 208
CF_WriteHistoryEntryToFile
 cf_utils.c, 1256
 cf_utils.h, 1278
CF_WriteHistoryQueueDataToFile
 cf_utils.c, 1257
 cf_utils.h, 1279
CF_WriteQueue_Payload, 657
 chan, 657
 filename, 657
 queue, 658
 spare, 658
 type, 658
CF_WriteQueue_Payload_t
 CFS CFDP Command Structures, 228
CF_WriteQueueCmd, 658
 cf_cmd.c, 1128
 cf_cmd.h, 1161
 CommandHeader, 658
 Payload, 658
CF_WriteQueueCmd_t
 CFS CFDP Command Structures, 229
CF_WriteTxnQueueDataToFile
 cf_utils.c, 1258
 cf_utils.h, 1280
cfdp_class
 CF_Playback, 629
 CF_PollDir, 632
 CF_TxFile_Payload, 652

- cFE Access Table Content APIs, [386](#)
 CFE_TBL_GetAddress, [386](#)
 CFE_TBL_GetAddresses, [387](#)
 CFE_TBL_ReleaseAddress, [388](#)
 CFE_TBL_ReleaseAddresses, [389](#)
- cFE Application Behavior APIs, [265](#)
 CFE_ES_ExitApp, [265](#)
 CFE_ES_IncrementTaskCounter, [265](#)
 CFE_ES_RunLoop, [266](#)
 CFE_ES_WaitForStartupSync, [267](#)
 CFE_ES_WaitForSystemState, [267](#)
- cFE Application Control APIs, [262](#)
 CFE_ES_DeleteApp, [262](#)
 CFE_ES_ReloadApp, [262](#)
 CFE_ES_RestartApp, [263](#)
- cFE Child Task APIs, [278](#)
 CFE_ES_CreateChildTask, [278](#)
 CFE_ES_DeleteChildTask, [279](#)
 CFE_ES_ExitChildTask, [280](#)
 CFE_ES_GetTaskIDByName, [280](#)
 CFE_ES_GetTaskName, [281](#)
- cFE Clock State Flag Defines, [417](#)
 CFE_TIME_FLAG_ADD1HZ, [417](#)
 CFE_TIME_FLAG_ADDADJ, [417](#)
 CFE_TIME_FLAG_ADDTCL, [417](#)
 CFE_TIME_FLAG_CLKSET, [417](#)
 CFE_TIME_FLAG_CMDFLY, [418](#)
 CFE_TIME_FLAG_FLYING, [418](#)
 CFE_TIME_FLAG_GDTONE, [418](#)
 CFE_TIME_FLAG_REFERR, [418](#)
 CFE_TIME_FLAG_SERVER, [418](#)
 CFE_TIME_FLAG_SIGPRI, [418](#)
 CFE_TIME_FLAG_SRCINT, [418](#)
 CFE_TIME_FLAG_SRVFLY, [418](#)
 CFE_TIME_FLAG_UNUSED, [418](#)
- cFE Critical Data Store APIs, [285](#)
 CFE_ES_CopyToCDS, [285](#)
 CFE_ES_GetCDSBlockIDByName, [286](#)
 CFE_ES_GetCDSBlockName, [286](#)
 CFE_ES_RegisterCDS, [287](#)
 CFE_ES_RestoreFromCDS, [288](#)
- cFE Entry/Exit APIs, [260](#)
 CFE_ES_Main, [260](#)
 CFE_ES_ResetCFE, [260](#)
- cFE External Time Source APIs, [407](#)
 CFE_TIME_ExternalGPS, [407](#)
 CFE_TIME_ExternalMET, [408](#)
 CFE_TIME_ExternalTime, [408](#)
 CFE_TIME_ExternalTone, [409](#)
 CFE_TIME_RegisterSynchCallback, [409](#)
 CFE_TIME_UnregisterSynchCallback, [410](#)
- cFE File Header Management APIs, [314](#)
 CFE_FS_InitHeader, [314](#)
 CFE_FS_ReadHeader, [314](#)
- CFE_FS_SetTimestamp, [315](#)
 CFE_FS_WriteHeader, [316](#)
- cFE File Utility APIs, [318](#)
 CFE_FS_BackgroundFileDumpIsPending, [318](#)
 CFE_FS_BackgroundFileDumpRequest, [319](#)
 CFE_FS_ExtractFilenameFromPath, [319](#)
 CFE_FS_GetDefaultExtension, [320](#)
 CFE_FS_GetDefaultMountPoint, [320](#)
 CFE_FS_ParseInputFileName, [320](#)
 CFE_FS_ParseInputFileNameEx, [321](#)
- cFE Generic Counter APIs, [299](#)
 CFE_ES_DeleteGenCounter, [299](#)
 CFE_ES_GetGenCount, [300](#)
 CFE_ES_GetGenCounterIDByName, [300](#)
 CFE_ES_GetGenCounterName, [301](#)
 CFE_ES_IncrementGenCounter, [302](#)
 CFE_ES_RegisterGenCounter, [302](#)
 CFE_ES_SetGenCount, [303](#)
- cFE Generic Message APIs, [323](#)
 CFE_MSG_Init, [323](#)
- cFE Get Current Time APIs, [396](#)
 CFE_TIME_GetMET, [396](#)
 CFE_TIME_GetMETseconds, [396](#)
 CFE_TIME_GetMETsubsecs, [397](#)
 CFE_TIME_GetTAI, [397](#)
 CFE_TIME_GetTime, [398](#)
 CFE_TIME_GetUTC, [398](#)
- cFE Get Table Information APIs, [391](#)
 CFE_TBL_GetInfo, [391](#)
 CFE_TBL_GetStatus, [392](#)
 CFE_TBL_NotifyByMessage, [392](#)
- cFE Get Time Information APIs, [399](#)
 CFE_TIME_GetClockInfo, [399](#)
 CFE_TIME_GetClockState, [399](#)
 CFE_TIME_GetLeapSeconds, [400](#)
 CFE_TIME_GetSTCF, [400](#)
- cFE Information APIs, [269](#)
 CFE_ES_GetAppID, [269](#)
 CFE_ES_GetAppIDByName, [270](#)
 CFE_ES_GetAppInfo, [270](#)
 CFE_ES_GetAppName, [271](#)
 CFE_ES_GetLibIDByName, [272](#)
 CFE_ES_GetLibInfo, [272](#)
 CFE_ES_GetLibName, [273](#)
 CFE_ES_GetModuleInfo, [274](#)
 CFE_ES_GetResetType, [275](#)
 CFE_ES_GetTaskID, [275](#)
 CFE_ES_GetTaskInfo, [276](#)
- cFE Manage Table Content APIs, [380](#)
 CFE_TBL_DumpToBuffer, [380](#)
 CFE_TBL_Load, [381](#)
 CFE_TBL_Manage, [382](#)
 CFE_TBL_Modified, [383](#)
 CFE_TBL_Update, [384](#)

CFE_TBL_Validate, 384
cFE Memory Manager APIs, 290
 CFE_ES_GetMemPoolStats, 290
 CFE_ES_GetPoolBuf, 291
 CFE_ES_GetPoolBufInfo, 291
 CFE_ES_PoolCreate, 292
 CFE_ES_PoolCreateEx, 293
 CFE_ES_PoolCreateNoSem, 294
 CFE_ES_PoolDelete, 295
 CFE_ES_PutPoolBuf, 296
cFE Message Characteristics APIs, 364
 CFE_SB_GetUserData, 364
 CFE_SB_GetUserDataLength, 364
 CFE_SB_MessageStringGet, 365
 CFE_SB_MessageStringSet, 366
 CFE_SB_SetUserDataLength, 367
 CFE_SB_TimeStampMsg, 367
cFE Message Extended Header APIs, 333
 CFE_MSG_GetEDSVersion, 333
 CFE_MSG_GetEndian, 334
 CFE_MSG_GetPlaybackFlag, 334
 CFE_MSG_GetSubsystem, 335
 CFE_MSG_GetSystem, 335
 CFE_MSG_SetEDSVersion, 336
 CFE_MSG_SetEndian, 336
 CFE_MSG_SetPlaybackFlag, 337
 CFE_MSG_SetSubsystem, 337
 CFE_MSG_SetSystem, 338
cFE Message ID APIs, 369
 CFE_SB_CmdTopicIdToMsgId, 369
 CFE_SB_GlobalCmdTopicIdToMsgId, 369
 CFE_SB_GlobalTlmTopicIdToMsgId, 370
 CFE_SB_IsValidMsgId, 370
 CFE_SB_LocalCmdTopicIdToMsgId, 371
 CFE_SB_LocalTlmTopicIdToMsgId, 371
 CFE_SB_MsgId_Equal, 371
 CFE_SB_MsgIdToValue, 372
 CFE_SB_TlmTopicIdToMsgId, 372
 CFE_SB_ValueToMsgId, 373
cFE Message Id APIs, 344
 CFE_MSG_GetMsgId, 344
 CFE_MSG_GetTypeFromMsgId, 344
 CFE_MSG_SetMsgId, 345
cFE Message Integrity APIs, 346
 CFE_MSG_OriginationAction, 346
 CFE_MSG_VerificationAction, 347
cFE Message Primary Header APIs, 324
 CFE_MSG_GetApld, 324
 CFE_MSG_GetHasSecondaryHeader, 325
 CFE_MSG_GetHeaderVersion, 325
 CFE_MSG_GetNextSequenceCount, 326
 CFE_MSG_GetSegmentationFlag, 326
 CFE_MSG_GetSequenceCount, 327
 CFE_MSG.GetSize, 327
 CFE_MSG_GetType, 328
 CFE_MSG_SetApld, 328
 CFE_MSG_SetHasSecondaryHeader, 329
 CFE_MSG_SetHeaderVersion, 329
 CFE_MSG_SetSegmentationFlag, 330
 CFE_MSG_SetSequenceCount, 330
 CFE_MSG_SetSize, 331
 CFE_MSG_SetType, 331
cFE Message Secondary Header APIs, 339
 CFE_MSG_GenerateChecksum, 339
 CFE_MSG_GetFcnCode, 340
 CFE_MSG_GetMsgTime, 340
 CFE_MSG_SetFcnCode, 341
 CFE_MSG_SetMsgTime, 341
 CFE_MSG_ValidateChecksum, 342
cFE Message Subscription Control APIs, 353
 CFE_SB_Subscribe, 353
 CFE_SB_SubscribeEx, 354
 CFE_SB_SubscribeLocal, 355
 CFE_SB_Unsubscribe, 355
 CFE_SB_UnsubscribeLocal, 356
cFE Miscellaneous APIs, 282
 CFE_ES_BackgroundWakeups, 282
 CFE_ES_CalculateCRC, 282
 CFE_ES_ProcessAsyncEvent, 283
 CFE_ES_WriteToSysLog, 283
cFE Miscellaneous Time APIs, 412
 CFE_TIME_Local1HzISR, 412
 CFE_TIME_Print, 412
cFE Performance Monitor APIs, 297
 CFE_ES_PerfLogAdd, 298
 CFE_ES_PerfLogEntry, 297
 CFE_ES_PerfLogExit, 297
cFE Pipe Management APIs, 348
 CFE_SB_CreatePipe, 348
 CFE_SB_DeletePipe, 349
 CFE_SB_GetPipeIdByName, 350
 CFE_SB_GetPipeName, 350
 CFE_SB_GetPipeOpts, 351
 CFE_SB_PipeId_ToIndex, 351
 CFE_SB_SetPipeOpts, 352
cFE Registration APIs, 305, 375
 CFE_EVS_Register, 305
 CFE_TBL_Register, 375
 CFE_TBL_Share, 377
 CFE_TBL_Unregister, 378
cFE Reset Event Filter APIs, 312
 CFE_EVS_ResetAllFilters, 312
 CFE_EVS_ResetFilter, 312
cFE Resource ID APIs, 257
 CFE_ES_AppID_ToIndex, 257
 CFE_ES_CounterID_ToIndex, 257
 CFE_ES_LibID_ToIndex, 258
 CFE_ES_TaskID_ToIndex, 259

cFE Resource ID base values, [415](#)
 CFE_CONFIGID_BASE, [416](#)
 CFE_ES_APPID_BASE, [416](#)
 CFE_ES_CDSBLOCKID_BASE, [416](#)
 CFE_ES_COUNTID_BASE, [416](#)
 CFE_ES_LIBID_BASE, [416](#)
 CFE_ES_POOLID_BASE, [416](#)
 CFE_ES_TASKID_BASE, [416](#)
 CFE_RESOURCEID_CONFIGID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_APPID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_LIBID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_POOLID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_ES_TASKID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET, [415](#)
 CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET, [415](#)
 CFE_SB_PIPEID_BASE, [416](#)
 CFE_TBL_DUMPCTRLID_BASE, [416](#)
 CFE_TBL_VALRESULTID_BASE, [416](#)

cFE Return Code Defines, [234](#)
 CFE_ES_APP_CLEANUP_ERR, [239](#)
 CFE_ES_BAD_ARGUMENT, [239](#)
 CFE_ES_BIN_SEM_DELETE_ERR, [239](#)
 CFE_ES_BUFFER_NOT_IN_POOL, [239](#)
 CFE_ES_CDS_ACCESS_ERROR, [239](#)
 CFE_ES_CDS_ALREADY_EXISTS, [240](#)
 CFE_ES_CDS_BLOCK_CRC_ERR, [240](#)
 CFE_ES_CDS_INSUFFICIENT_MEMORY, [240](#)
 CFE_ES_CDS_INVALID, [240](#)
 CFE_ES_CDS_INVALID_NAME, [240](#)
 CFE_ES_CDS_INVALID_SIZE, [240](#)
 CFE_ES_CDS_OWNER_ACTIVE_ERR, [240](#)
 CFE_ES_CDS_WRONG_TYPE_ERR, [240](#)
 CFE_ES_COUNT_SEM_DELETE_ERR, [241](#)
 CFE_ES_ERR_APP_CREATE, [241](#)
 CFE_ES_ERR_APP_REGISTER, [241](#)
 CFE_ES_ERR_CHILD_TASK_CREATE, [241](#)
 CFE_ES_ERR_CHILD_TASK_DELETE, [241](#)
 CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK, [241](#)
 CFE_ES_ERR_CHILD_TASK_REGISTER, [241](#)
 CFE_ES_ERR_DUPLICATE_NAME, [241](#)

CFE_ES_ERR_LOAD_LIB, [242](#)
 CFE_ES_ERR_MEM_BLOCK_SIZE, [242](#)
 CFE_ES_ERR_NAME_NOT_FOUND, [242](#)
 CFE_ES_ERR_RESOURCEID_NOT_VALID, [242](#)
 CFE_ES_ERR_SYS_LOG_FULL, [242](#)
 CFE_ES_ERR_SYS_LOG_TRUNCATED, [242](#)
 CFE_ES_FILE_CLOSE_ERR, [242](#)
 CFE_ES_FILE_IO_ERR, [242](#)
 CFE_ES_LIB_ALREADY_LOADED, [243](#)
 CFE_ES_MUT_SEM_DELETE_ERR, [243](#)
 CFE_ES_NO_RESOURCE_IDS_AVAILABLE, [243](#)
 CFE_ES_NOT_IMPLEMENTED, [243](#)
 CFE_ES_OPERATION_TIMED_OUT, [243](#)
 CFE_ES_POOL_BLOCK_INVALID, [243](#)
 CFE_ES_QUEUE_DELETE_ERR, [243](#)
 CFE_ES_RST_ACCESS_ERR, [243](#)
 CFE_ES_TASK_DELETE_ERR, [243](#)
 CFE_ES_TIMER_DELETE_ERR, [244](#)
 CFE_EVS_APP_FILTER_OVERLOAD, [244](#)
 CFE_EVS_APP_ILLEGAL_APP_ID, [244](#)
 CFE_EVS_APP_NOT_REGISTERED, [244](#)
 CFE_EVS_APP_SQUELCHED, [244](#)
 CFE_EVS_EVT_NOT_REGISTERED, [244](#)
 CFE_EVS_FILE_WRITE_ERROR, [244](#)
 CFE_EVS_INVALID_PARAMETER, [244](#)
 CFE_EVS_NOT_IMPLEMENTED, [245](#)
 CFE_EVS_RESET_AREA_POINTER, [245](#)
 CFE_EVS_UNKNOWN_FILTER, [245](#)
 CFE_FS_BAD_ARGUMENT, [245](#)
 CFE_FS_FNAME_TOO_LONG, [245](#)
 CFE_FS_INVALID_PATH, [245](#)
 CFE_FS_NOT_IMPLEMENTED, [245](#)
 CFE_SB_BAD_ARGUMENT, [245](#)
 CFE_SB_BUF_ALOC_ERR, [245](#)
 CFE_SB_BUFFER_INVALID, [246](#)
 CFE_SB_INTERNAL_ERR, [246](#)
 CFE_SB_MAX_DESTS_MET, [246](#)
 CFE_SB_MAX_MSGS_MET, [246](#)
 CFE_SB_MAX_PIPES_MET, [246](#)
 CFE_SB_MSG_TOO_BIG, [246](#)
 CFE_SB_NO_MESSAGE, [246](#)
 CFE_SB_NOT_IMPLEMENTED, [247](#)
 CFE_SB_PIPE_CR_ERR, [247](#)
 CFE_SB_PIPE_RD_ERR, [247](#)
 CFE_SB_TIME_OUT, [247](#)
 CFE_SB_WRONG_MSG_TYPE, [247](#)
 CFE_STATUS_BAD_COMMAND_CODE, [247](#)
 CFE_STATUS_EXTERNAL_RESOURCE_FAIL, [247](#)
 CFE_STATUS_INCORRECT_STATE, [248](#)
 CFE_STATUS_NO_COUNTER_INCREMENT, [248](#)
 CFE_STATUS_NOT_IMPLEMENTED, [248](#)
 CFE_STATUS_RANGE_ERROR, [248](#)
 CFE_STATUS_REQUEST_ALREADY_PENDING, [248](#)

- CFE_STATUS_UNKNOWN_MSG_ID, 248
CFE_STATUS_VALIDATION_FAILURE, 248
CFE_STATUS_WRONG_MSG_LENGTH, 249
CFE_SUCCESS, 249
CFE_TBL_BAD_ARGUMENT, 249
CFE_TBL_ERR_ACCESS, 249
CFE_TBL_ERR_BAD_CONTENT_ID, 249
CFE_TBL_ERR_BAD_PROCESSOR_ID, 249
CFE_TBL_ERR_BAD_SPACECRAFT_ID, 249
CFE_TBL_ERR_BAD_SUBTYPE_ID, 249
CFE_TBL_ERR_DUMP_ONLY, 250
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE, 250
CFE_TBL_ERR_DUPLICATE_NOT OWNED, 250
CFE_TBL_ERR_FILE_FOR_WRONG_TABLE, 250
CFE_TBL_ERR_FILE_SIZE_INCONSISTENT, 250
CFE_TBL_ERR_FILE_TOO_LARGE, 250
CFE_TBL_ERR_FILENAME_TOO_LONG, 250
CFE_TBL_ERR_HANDLES_FULL, 250
CFE_TBL_ERR_ILLEGAL_SRC_TYPE, 251
CFE_TBL_ERR_INVALID_HANDLE, 251
CFE_TBL_ERR_INVALID_NAME, 251
CFE_TBL_ERR_INVALID_OPTIONS, 251
CFE_TBL_ERR_INVALID_SIZE, 251
CFE_TBL_ERR_LOAD_IN_PROGRESS, 251
CFE_TBL_ERR_LOAD_INCOMPLETE, 252
CFE_TBL_ERR_NEVER_LOADED, 252
CFE_TBL_ERR_NO_ACCESS, 252
CFE_TBL_ERR_NO_BUFFER_AVAIL, 252
CFE_TBL_ERR_NO_STD_HEADER, 252
CFE_TBL_ERR_NO_TBL_HEADER, 252
CFE_TBL_ERR_PARTIAL_LOAD, 252
CFE_TBL_ERR_REGISTRY_FULL, 252
CFE_TBL_ERR_SHORT_FILE, 253
CFE_TBL_ERR_UNREGISTERED, 253
CFE_TBL_INFO_DUMP_PENDING, 253
CFE_TBL_INFO_NO_UPDATE_PENDING, 253
CFE_TBL_INFO_NO_VALIDATION_PENDING, 253
CFE_TBL_INFO_RECOVERED_TBL, 253
CFE_TBL_INFO_TABLE_LOCKED, 253
CFE_TBL_INFO_UPDATE_PENDING, 253
CFE_TBL_INFO_UPDATED, 254
CFE_TBL_INFO_VALIDATION_PENDING, 254
CFE_TBL_MESSAGE_ERROR, 254
CFE_TBL_NOT_IMPLEMENTED, 254
CFE_TBL_WARN_DUPLICATE, 254
CFE_TBL_WARN_NOT_CRITICAL, 254
CFE_TBL_WARN_PARTIAL_LOAD, 254
CFE_TBL_WARN_SHORT_FILE, 255
CFE_TIME_BAD_ARGUMENT, 255
CFE_TIME_CALLBACK_NOT_REGISTERED, 255
CFE_TIME_INTERNAL_ONLY, 255
CFE_TIME_NOT_IMPLEMENTED, 255
CFE_TIME_OUT_OF_RANGE, 255
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS, 255
cFE SB Pipe options, 374
CFE_SB_PIPEOPTS_IGNOREMINE, 374
cFE Send Event APIs, 307
CFE_EVS_SendEvent, 307
CFE_EVS_SendEventWithAppID, 308
CFE_EVS_SendTimedEvent, 309
cFE Send/Receive Message APIs, 358
CFE_SB_ReceiveBuffer, 358
CFE_SB_TransmitMsg, 359
cFE Table Type Defines, 394
CFE_TBL_OPT_BUFFER_MSK, 394
CFE_TBL_OPT_CRITICAL, 394
CFE_TBL_OPT_CRITICAL_MSK, 394
CFE_TBL_OPT_DBL_BUFFER, 394
CFE_TBL_OPT_DEFAULT, 395
CFE_TBL_OPT_DUMP_ONLY, 395
CFE_TBL_OPT_LD_DMP_MSK, 395
CFE_TBL_OPT_LOAD_DUMP, 395
CFE_TBL_OPT_NOT_CRITICAL, 395
CFE_TBL_OPT_NOT_USR_DEF, 395
CFE_TBL_OPT_SNGL_BUFFER, 395
CFE_TBL_OPT_USR_DEF_ADDR, 395
CFE_TBL_OPT_USR_DEF_MSK, 395
cFE Time Arithmetic APIs, 402
CFE_TIME_Add, 402
CFE_TIME_Compare, 402
CFE_TIME_Subtract, 403
cFE Time Conversion APIs, 405
CFE_TIME_MET2SCTime, 405
CFE_TIME_Micro2SubSecs, 405
CFE_TIME_Sub2MicroSecs, 406
cFE Zero Copy APIs, 361
CFE_SB_AllocateMessageBuffer, 361
CFE_SB_ReleaseMessageBuffer, 361
CFE_SB_TransmitBuffer, 362
cfe/cmake/sample_defs/example_mission_cfg.h, 1287
cfe/cmake/sample_defs/example_platform_cfg.h, 1298
cfe/cmake/sample_defs/sample_perfids.h, 1342
cfe/docs/src/cfe_api.dox, 1344
cfe/docs/src/cfe_es.dox, 1344
cfe/docs/src/cfe_evs.dox, 1344
cfe/docs/src/cfe_frontpage.dox, 1344
cfe/docs/src/cfe_glossary.dox, 1344
cfe/docs/src/cfe_sb.dox, 1344
cfe/docs/src/cfe_tbl.dox, 1344
cfe/docs/src/cfe_time.dox, 1344
cfe/docs/src/cfe_xref.dox, 1344
cfe/docs/src/cfs_versions.dox, 1344
cfe/modules/config/fsw/inc/cfe_config_external.h, 1344
cfe/modules/config/fsw/inc/cfe_config_init.h, 1345
cfe/modules/config/fsw/inc/cfe_config_lookup.h, 1345
cfe/modules/config/fsw/inc/cfe_config_nametable.h, 1346

cfe/modules/config/fsw/inc/cfe_config_set.h, 1346
 cfe/modules/config/fsw/inc/cfe_config_table.h, 1347
 cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h, 1348
 cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h, 1350
 cfe/modules/core_api/config/default_cfe_mission_cfg.h, 1352
 cfe/modules/core_api/config/default_cfe_msgids.h, 1353
 cfe/modules/core_api/fsw/inc/cfe.h, 1353
 cfe/modules/core_api/fsw/inc/cfe_config.h, 1353
 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h, 1357
 cfe/modules/core_api/fsw/inc/cfe_endian.h, 1358
 cfe/modules/core_api/fsw/inc/cfe_error.h, 1359
 cfe/modules/core_api/fsw/inc/cfe_es.h, 1367
 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h, 1371
 cfe/modules/core_api/fsw/inc/cfe_evs.h, 1376
 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h, 1377
 cfe/modules/core_api/fsw/inc/cfe_fs.h, 1380
 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h, 1380
 cfe/modules/core_api/fsw/inc/cfe_msg.h, 1383
 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h, 1385
 cfe/modules/core_api/fsw/inc/cfe_resourceid.h, 1389
 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h, 1396
 cfe/modules/core_api/fsw/inc/cfe_sb.h, 1396
 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h, 1399
 cfe/modules/core_api/fsw/inc/cfe_tbl.h, 1402
 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h, 1403
 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h, 1406
 cfe/modules/core_api/fsw/inc/cfe_time.h, 1407
 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h, 1409
 cfe/modules/core_api/fsw/inc/cfe_version.h, 1411
 cfe/modules/es/config/default_cfe_es_extern_typedefs.h, 1413
 cfe/modules/es/config/default_cfe_es_fcncodes.h, 1421
 cfe/modules/es/config/default_cfe_es_interface_cfg.h, 1442
 cfe/modules/es/config/default_cfe_es_internal_cfg.h, 1445
 cfe/modules/es/config/default_cfe_es_mission_cfg.h, 1466
 cfe/modules/es/config/default_cfe_es_msg.h, 1466
 cfe/modules/es/config/default_cfe_es_msgdefs.h, 1466
 cfe/modules/es/config/default_cfe_es_msgids.h, 1470
 cfe/modules/es/config/default_cfe_es_msgstruct.h, 1471
 cfe/modules/es/config/default_cfe_es_platform_cfg.h, 1475
 cfe/modules/es/config/default_cfe_es_topicids.h, 1476
 cfe/modules/es/fsw/inc/cfe_es_eventids.h, 1477
 cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h, 1502
 cfe/modules/evs/config/default_cfe_evs_fcncodes.h, 1505
 cfe/modules/evs/config/default_cfe_evs_interface_cfg.h, 1523
 cfe/modules/evs/config/default_cfe_evs_internal_cfg.h, 1524
 cfe/modules/evs/config/default_cfe_evs_mission_cfg.h, 1528
 cfe/modules/evs/config/default_cfe_evs_msg.h, 1528
 cfe/modules/evs/config/default_cfe_evs_msgdefs.h, 1528
 cfe/modules/evs/config/default_cfe_evs_msgids.h, 1532
 cfe/modules/evs/config/default_cfe_evs_msgstruct.h, 1533
 cfe/modules/evs/config/default_cfe_evs_platform_cfg.h, 1536
 cfe/modules/evs/config/default_cfe_evs_topicids.h, 1536
 cfe/modules/evs/fsw/inc/cfe_evs_eventids.h, 1537
 cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h, 1549
 cfe/modules/fs/config/default_cfe_fs_filedef.h, 1549
 cfe/modules/fs/config/default_cfe_fs_interface_cfg.h, 1551
 cfe/modules/fs/config/default_cfe_fs_mission_cfg.h, 1552
 cfe/modules/msg/fsw/inc/ccsds_hdr.h, 1552
 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h, 1553
 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h, 1553
 cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h, 1554
 cfe/modules/sb/config/default_cfe_sb_fcncodes.h, 1556
 cfe/modules/sb/config/default_cfe_sb_interface_cfg.h, 1566
 cfe/modules/sb/config/default_cfe_sb_internal_cfg.h, 1567
 cfe/modules/sb/config/default_cfe_sb_mission_cfg.h, 1575
 cfe/modules/sb/config/default_cfe_sb_msg.h, 1575
 cfe/modules/sb/config/default_cfe_sb_msgdefs.h, 1576
 cfe/modules/sb/config/default_cfe_sb_msgids.h, 1578
 cfe/modules/sb/config/default_cfe_sb_msgstruct.h, 1580
 cfe/modules/sb/config/default_cfe_sb_platform_cfg.h, 1582
 cfe/modules/sb/config/default_cfe_sb_topicids.h, 1582
 cfe/modules/sb/fsw/inc/cfe_sb_eventids.h, 1583
 cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h, 1603
 cfe/modules/tbl/config/default_cfe_tbl_fcncodes.h, 1604
 cfe/modules/tbl/config/default_cfe_tbl_interface_cfg.h, 1613
 cfe/modules/tbl/config/default_cfe_tbl_internal_cfg.h, 1614

cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h, 1620
cfe/modules/tbl/config/default_cfe_tbl_msg.h, 1620
cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h, 1621
cfe/modules/tbl/config/default_cfe_tbl_msgids.h, 1623
cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h, 1624
cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h, 1626
cfe/modules/tbl/config/default_cfe_tbl_topicids.h, 1626
cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h, 1627
cfe/modules/time/config/default_cfe_time_extern_typedefs.h, CFE_Config_Callback_t, 1358
cfe/modules/time/config/default_cfe_time_fcncodes.h, 1652
cfe/modules/time/config/default_cfe_time_interface_cfg.h, 1668
cfe/modules/time/config/default_cfe_time_internal_cfg.h, 1672
cfe/modules/time/config/default_cfe_time_mission_cfg.h, 1677
cfe/modules/time/config/default_cfe_time_msg.h, 1678
cfe/modules/time/config/default_cfe_time_msgdefs.h, 1678
cfe/modules/time/config/default_cfe_time_msgids.h, 1680
cfe/modules/time/config/default_cfe_time_msgstruct.h, 1682
cfe/modules/time/config/default_cfe_time_platform_cfg.h, 1685
cfe/modules/time/config/default_cfe_time_topicids.h, 1685
cfe/modules/time/fsw/inc/cfe_time_eventids.h, 1687
CFE_BIT
 cfe_sb.h, 1398
CFE_BUILD_BASELINE
 cfe_version.h, 1411
CFE_BUILD_CODENAME
 cfe_version.h, 1411
CFE_BUILD_DEV_CYCLE
 cfe_version.h, 1411
CFE_BUILD_NUMBER
 cfe_version.h, 1412
CFE_CFG_MAX_VERSION_STR_LEN
 cfe_version.h, 1412
CFE_CLR
 cfe_sb.h, 1399
cfe_config.h
 CFE_Config_GetArrayValue, 1354
 CFE_Config_GetIdByName, 1354
 CFE_Config_getName, 1355
 CFE_Config_GetObjPointer, 1355
 CFE_Config_GetString, 1355
 CFE_Config_GetValue, 1356
 CFE_Config_GetVersionString, 1356
 CFE_Config_IterateAll, 1357
cfe_config_api_typedefs.h
 CFE_Config_ArrayValue_t, 1358
 CFE_Config_Callback_t, 1358
 CFE_CONFIGID_C, 1358
 CFE_ConfigId_t, 1358
 CFE_CONFIGID_UNDEFINED, 1358
CFE_Config_ArrayValue, 659
 ElementPtr, 659
 NumElements, 659
CFE_Config_ArrayValue_t
 cfe_config_api_typedefs.h, 1358
CFE_Config_Callback_t
 cfe_config_api_typedefs.h, 1358
cfe_config_external.h
 CFE_Config_SetupPlatformConfigInfo, 1345
CFE_Config_GetArrayValue
 cfe_config.h, 1354
CFE_Config_GetIdByName
 cfe_config.h, 1354
CFE_Config_getName
 cfe_config.h, 1355
CFE_Config_GetObjPointer
 cfe_config.h, 1355
CFE_Config_GetString
 cfe_config.h, 1355
CFE_Config_GetValue
 cfe_config.h, 1356
CFE_Config_GetVersionString
 cfe_config.h, 1356
CFE_Config_IdNameEntry, 659
 Name, 659
CFE_Config_IdNameEntry_t
 cfe_config_nametable.h, 1346
CFE_Config_Init
 cfe_config_init.h, 1345
cfe_config_init.h
 CFE_Config_Init, 1345
 CFE_Config_SetupBasicBuildInfo, 1345
CFE_Config_IterateAll
 cfe_config.h, 1357
CFE_Config_LocateConfigRecordByID
 cfe_config_lookup.h, 1346
cfe_config_lookup.h
 CFE_Config_LocateConfigRecordByID, 1346
cfe_config_nametable.h
 CFE_Config_IdNameEntry_t, 1346
 CFE_CONFIGID_NAMETABLE, 1346
cfe_config_set.h
 CFE_Config_SetArrayValue, 1347
 CFE_Config_SetObjPointer, 1347
 CFE_Config_SetString, 1347
 CFE_Config_SetValue, 1347
CFE_Config_SetArrayValue
 cfe_config_set.h, 1347
CFE_Config_SetObjPointer

cfe_config_set.h, 1347
 CFE_Config_SetString
 cfe_config_set.h, 1347
 CFE_Config_SetupBasicBuildInfo
 cfe_config_init.h, 1345
 CFE_Config_SetupPlatformConfigInfo
 cfe_config_external.h, 1345
 CFE_Config_SetValue
 cfe_config_set.h, 1347
 cfe_config_table.h
 CFE_Config_ValueBuffer_t, 1348
 CFE_Config_ValueEntry_t, 1348
 CFE_ConfigType, 1348
 CFE_ConfigType_ARRAY, 1348
 CFE_ConfigType_POINTER, 1348
 CFE_ConfigType_STRING, 1348
 CFE_ConfigType_t, 1348
 CFE_ConfigType_UNDEFINED, 1348
 CFE_ConfigType_VALUE, 1348
 CFE_Config_ValueBuffer, 660
 AsInteger, 660
 AsPointer, 660
 CFE_Config_ValueBuffer_t
 cfe_config_table.h, 1348
 CFE_Config_ValueEntry, 660
 ActualType, 660
 Datum, 660
 CFE_Config_ValueEntry_t
 cfe_config_table.h, 1348
 CFE_CONFIGID_BASE
 cFE Resource ID base values, 416
 CFE_CONFIGID_C
 cfe_config_api_typedefs.h, 1358
 CFE_CONFIGID_NAMETABLE
 cfe_config_nametable.h, 1346
 CFE_ConfigId_t
 cfe_config_api_typedefs.h, 1358
 CFE_CONFIGID_UNDEFINED
 cfe_config_api_typedefs.h, 1358
 CFE_ConfigType
 cfe_config_table.h, 1348
 CFE_ConfigType_ARRAY
 cfe_config_table.h, 1348
 CFE_ConfigType_POINTER
 cfe_config_table.h, 1348
 CFE_ConfigType_STRING
 cfe_config_table.h, 1348
 CFE_ConfigType_t
 cfe_config_table.h, 1348
 CFE_ConfigType_UNDEFINED
 cfe_config_table.h, 1348
 CFE_ConfigType_VALUE
 cfe_config_table.h, 1348
 CFE_CPU1_CMD_MID_BASE
 default_cfe_core_api_base_msgids.h, 1349
 CFE_CPU1_TLM_MID_BASE
 default_cfe_core_api_base_msgids.h, 1349
 cfe_endian.h
 CFE_MAKE_BIG16, 1359
 CFE_MAKE_BIG32, 1359
 cfe_error.h
 CFE_ES_StatusToString, 1367
 CFE_EVENTS_SERVICE, 1365
 CFE_EXECUTIVE_SERVICE, 1365
 CFE_FILE_SERVICE, 1365
 CFE_GENERIC_SERVICE, 1366
 CFE_SERVICE_BITMASK, 1366
 CFE_SEVERITY_BITMASK, 1366
 CFE_SEVERITY_ERROR, 1366
 CFE_SEVERITY_INFO, 1366
 CFE_SEVERITY_SUCCESS, 1366
 CFE_SOFTWARE_BUS_SERVICE, 1366
 CFE_STATUS_C, 1366
 CFE_STATUS_STRING_LENGTH, 1366
 CFE_Status_t, 1367
 CFE_StatusString_t, 1367
 CFE_TABLE_SERVICE, 1366
 CFE_TIME_SERVICE, 1367
 cfe_es.h
 CFE_ES_DBIT, 1370
 CFE_ES_DTEST, 1370
 CFE_ES_TEST_LONG_MASK, 1371
 OS_PRINTF, 1371
 CFE_ES_ALL_APPS_EID
 cfe_es_eventids.h, 1480
 cfe_es_api_typedefs.h
 CFE_ES_APP_RESTART, 1372
 CFE_ES_APPID_C, 1372
 CFE_ES_APPID_UNDEFINED, 1373
 CFE_ES_CDS_BAD_HANDLE, 1373
 CFE_ES_CDSHANDLE_C, 1373
 CFE_ES_ChildTaskMainFuncPtr_t, 1374
 CFE_ES_COUNTERID_C, 1373
 CFE_ES_COUNTERID_UNDEFINED, 1373
 CFE_ES_CrcType_16_ARC, 1376
 CFE_ES_CrcType_CRC_16, 1376
 CFE_ES_CrcType_CRC_32, 1376
 CFE_ES_CrcType_CRC_8, 1376
 CFE_ES_CrcType_Enum, 1375
 CFE_ES_CrcType_Enum_t, 1374
 CFE_ES_CrcType_MAX, 1376
 CFE_ES_CrcType_NONE, 1376
 CFE_ES_LIBID_C, 1373
 CFE_ES_LIBID_UNDEFINED, 1373
 CFE_ES_LibraryEntryFuncPtr_t, 1375
 CFE_ES_MEMHANDLE_C, 1373
 CFE_ES_MEMHANDLE_UNDEFINED, 1373
 CFE_ES_MEMPOOLBUF_C, 1373

CFE_ES_MemPoolBuf_t, 1375
CFE_ES_NO_MUTEX, 1374
CFE_ES_PoolAlign_t, 1375
CFE_ES_StackPointer_t, 1375
CFE_ES_STATIC_POOL_TYPE, 1374
CFE_ES_TASK_STACK_ALLOCATE, 1374
CFE_ES_TaskEntryFuncPtr_t, 1375
CFE_ES_TASKID_C, 1374
CFE_ES_TASKID_UNDEFINED, 1374
CFE_ES_USE_MUTEX, 1374
CFE_ES_APP_CLEANUP_ERR
 cFE Return Code Defines, 239
CFE_ES_APP_RESTART
 cfe_es_api_typedefs.h, 1372
CFE_ES_APP_TLM_MID
 default_cfe_es_msgids.h, 1470
CFE_ES_APPID_BASE
 cFE Resource ID base values, 416
CFE_ES_APPID_C
 cfe_es_api_typedefs.h, 1372
CFE_ES_AppId_t
 default_cfe_es_extern_typedefs.h, 1416
CFE_ES_AppId_ToIndex
 cFE Resource ID APIs, 257
CFE_ES_APPID_UNDEFINED
 cfe_es_api_typedefs.h, 1373
CFE_ES_AppInfo, 661
 AddressesAreValid, 662
 BSSAddress, 662
 BSSSize, 662
 CodeAddress, 662
 CodeSize, 662
 DataAddress, 662
 DataSize, 663
 EntryPoint, 663
 ExceptionAction, 663
 ExecutionCounter, 663
 FileName, 663
 MainTaskId, 663
 MainTaskName, 663
 Name, 664
 NumOfChildTasks, 664
 Priority, 664
 ResourceId, 664
 StackSize, 664
 StartAddress, 664
 Type, 664
CFE_ES_AppInfo_t
 default_cfe_es_extern_typedefs.h, 1416
CFE_ES_AppNameCmd_Payload, 665
 Application, 665
CFE_ES_AppNameCmd_Payload_t
 default_cfe_es_msgdefs.h, 1468
CFE_ES_AppReloadCmd_Payload, 665
 AppFileName, 666
 Application, 666
CFE_ES_AppReloadCmd_Payload_t
 default_cfe_es_msgdefs.h, 1468
CFE_ES_AppState
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_EARLY_INIT
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_Enum_t
 default_cfe_es_extern_typedefs.h, 1416
CFE_ES_AppState_LATE_INIT
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_MAX
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_RUNNING
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_STOPPED
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_UNDEFINED
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppState_WAITING
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppType
 default_cfe_es_extern_typedefs.h, 1419
CFE_ES_AppType_CORE
 default_cfe_es_extern_typedefs.h, 1420
CFE_ES_AppType_Enum_t
 default_cfe_es_extern_typedefs.h, 1416
CFE_ES_AppType_EXTERNAL
 default_cfe_es_extern_typedefs.h, 1420
CFE_ES_AppType_LIBRARY
 default_cfe_es_extern_typedefs.h, 1420
CFE_ES_BackgroundWakeups
 cFE Miscellaneous APIs, 282
CFE_ES_BAD_ARGUMENT
 cFE Return Code Defines, 239
CFE_ES_BIN_SEM_DELETE_ERR
 cFE Return Code Defines, 239
CFE_ES_BlockStats, 666
 BlockSize, 666
 NumCreated, 666
 NumFree, 666
CFE_ES_BlockStats_t
 default_cfe_es_extern_typedefs.h, 1416
CFE_ES_BOOT_ERR_EID
 cfe_es_eventids.h, 1480
CFE_ES_BUFFER_NOT_IN_POOL
 cFE Return Code Defines, 239
CFE_ES_BUILD_INF_EID
 cfe_es_eventids.h, 1480
CFE_ES_CalculateCRC
 cFE Miscellaneous APIs, 282
CFE_ES_CC1_ERR_EID
 cfe_es_eventids.h, 1481

CFE_ES_CDS_ACCESS_ERROR
 cFE Return Code Defines, [239](#)
CFE_ES_CDS_ALREADY_EXISTS
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_BAD_HANDLE
cfe_es_api_typedefs.h, [1373](#)
CFE_ES_CDS_BLOCK_CRC_ERR
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_DELETE_ERR_EID
cfe_es_eventids.h, [1481](#)
CFE_ES_CDS_DELETE_TBL_ERR_EID
cfe_es_eventids.h, [1481](#)
CFE_ES_CDS_DELETED_INFO_EID
cfe_es_eventids.h, [1482](#)
CFE_ES_CDS_DUMP_ERR_EID
cfe_es_eventids.h, [1482](#)
CFE_ES_CDS_INSUFFICIENT_MEMORY
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_INVALID
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_INVALID_NAME
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_INVALID_SIZE
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_NAME_ERR_EID
cfe_es_eventids.h, [1482](#)
CFE_ES_CDS_OWNER_ACTIVE_EID
cfe_es_eventids.h, [1482](#)
CFE_ES_CDS_OWNER_ACTIVE_ERR
 cFE Return Code Defines, [240](#)
CFE_ES_CDS_REG_DUMP_INF_EID
cfe_es_eventids.h, [1483](#)
CFE_ES_CDS_REGISTER_ERR_EID
cfe_es_eventids.h, [1483](#)
CFE_ES_CDS_WRONG_TYPE_ERR
 cFE Return Code Defines, [240](#)
CFE_ES_CDSBLOCKID_BASE
 cFE Resource ID base values, [416](#)
CFE_ES_CDSHANDLE_C
cfe_es_api_typedefs.h, [1373](#)
CFE_ES_CDCHandle_t
default_cfe_es_extern_typedefs.h, [1416](#)
CFE_ES_CDSRegDumpRec, [667](#)

- ByteAlignSpare, [667](#)
- Handle, [667](#)
- Name, [667](#)
- Size, [668](#)
- Table, [668](#)

CFE_ES_CDSRegDumpRec_t
default_cfe_es_extern_typedefs.h, [1416](#)
CFE_ES_ChildTaskMainFuncPtr_t
cfe_es_api_typedefs.h, [1374](#)
CFE_ES_CLEAR_ER_LOG_CC
default_cfe_es_fcncodes.h, [1422](#)
CFE_ES_CLEAR_SYS_LOG_CC
default_cfe_es_fcncodes.h, [1422](#)
CFE_ES_ClearERLogCmd, [668](#)

- CommandHeader, [668](#)

CFE_ES_ClearERLogCmd_t
default_cfe_es_msgstruct.h, [1473](#)
CFE_ES_ClearSysLogCmd, [668](#)

- CommandHeader, [669](#)

CFE_ES_ClearSysLogCmd_t
default_cfe_es_msgstruct.h, [1473](#)
CFE_ES_CMD_MID
default_cfe_es_msgids.h, [1470](#)
CFE_ES_CopyToCDS
 cFE Critical Data Store APIs, [285](#)
CFE_ES_COUNT_SEM_DELETE_ERR
 cFE Return Code Defines, [241](#)
CFE_ES_COUNTERID_C
cfe_es_api_typedefs.h, [1373](#)
CFE_ES_CounterId_t
default_cfe_es_extern_typedefs.h, [1417](#)
CFE_ES_CounterID_ToIndex
 cFE Resource ID APIs, [257](#)
CFE_ES_COUNTERID_UNDEFINED
cfe_es_api_typedefs.h, [1373](#)
CFE_ES_COUNTID_BASE
 cFE Resource ID base values, [416](#)
CFE_ES_CrcType_16_ARC
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CrcType_CRC_16
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CrcType_CRC_32
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CrcType_CRC_8
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CrcType_Enum
cfe_es_api_typedefs.h, [1375](#)
CFE_ES_CrcType_Enum_t
cfe_es_api_typedefs.h, [1374](#)
CFE_ES_CrcType_MAX
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CrcType_NONE
cfe_es_api_typedefs.h, [1376](#)
CFE_ES_CreateChildTask
 cFE Child Task APIs, [278](#)
CFE_ES_CREATING_CDS_DUMP_ERR_EID
cfe_es_eventids.h, [1483](#)
CFE_ES_DBIT
cfe_es.h, [1370](#)
CFE_ES_DELETE_CDS_CC
default_cfe_es_fcncodes.h, [1423](#)
CFE_ES_DeleteApp
 cFE Application Control APIs, [262](#)
CFE_ES_DeleteCDSCmd, [669](#)

- CommandHeader, [669](#)

Payload, 669
CFE_ES_DeleteCDSCmd_Payload, 669
 CdsName, 670
CFE_ES_DeleteCDSCmd_Payload_t
 default_cfe_es_msgdefs.h, 1468
CFE_ES_DeleteCDSCmd_t
 default_cfe_es_msgstruct.h, 1473
CFE_ES_DeleteChildTask
 cFE Child Task APIs, 279
CFE_ES_DeleteGenCounter
 cFE Generic Counter APIs, 299
CFE_ES_DTEST
 cfe_es.h, 1370
CFE_ES_DUMP_CDS_REGISTRY_CC
 default_cfe_es_fcncodes.h, 1424
CFE_ES_DumpCDSRegistryCmd, 670
 CommandHeader, 670
 Payload, 670
CFE_ES_DumpCDSRegistryCmd_Payload, 670
 DumpFilename, 671
CFE_ES_DumpCDSRegistryCmd_Payload_t
 default_cfe_es_msgdefs.h, 1468
CFE_ES_DumpCDSRegistryCmd_t
 default_cfe_es_msgstruct.h, 1473
CFE_ES_ERLOG1_INF_EID
 cfe_es_eventids.h, 1483
CFE_ES_ERLOG2_EID
 cfe_es_eventids.h, 1484
CFE_ES_ERLOG2_ERR_EID
 cfe_es_eventids.h, 1484
CFE_ES_ERLOG_PENDING_ERR_EID
 cfe_es_eventids.h, 1484
CFE_ES_ERR_APP_CREATE
 cFE Return Code Defines, 241
CFE_ES_ERR_APP_REGISTER
 cFE Return Code Defines, 241
CFE_ES_ERR_CHILD_TASK_CREATE
 cFE Return Code Defines, 241
CFE_ES_ERR_CHILD_TASK_DELETE
 cFE Return Code Defines, 241
CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK
 cFE Return Code Defines, 241
CFE_ES_ERR_CHILD_TASK_REGISTER
 cFE Return Code Defines, 241
CFE_ES_ERR_DUPLICATE_NAME
 cFE Return Code Defines, 241
CFE_ES_ERR_LOAD_LIB
 cFE Return Code Defines, 242
CFE_ES_ERR_MEM_BLOCK_SIZE
 cFE Return Code Defines, 242
CFE_ES_ERR_NAME_NOT_FOUND
 cFE Return Code Defines, 242
CFE_ES_ERR_RESOURCEID_NOT_VALID
 cFE Return Code Defines, 242
CFE_ES_ERR_SYS_LOG_FULL
 cFE Return Code Defines, 242
CFE_ES_ERR_SYS_LOG_TRUNCATED
 cFE Return Code Defines, 242
CFE_ES_ERR_SYSLOGMODE_EID
 cfe_es_eventids.h, 1484
CFE_ES_ERREXIT_APP_ERR_EID
 cfe_es_eventids.h, 1485
CFE_ES_ERREXIT_APP_INF_EID
 cfe_es_eventids.h, 1485
cfe_es_eventids.h
 CFE_ES_ALL_APPS_EID, 1480
 CFE_ES_BOOT_ERR_EID, 1480
 CFE_ES_BUILD_INF_EID, 1480
 CFE_ES_CC1_ERR_EID, 1481
 CFE_ES_CDS_DELETE_ERR_EID, 1481
 CFE_ES_CDS_DELETE_TBL_ERR_EID, 1481
 CFE_ES_CDS_DELETED_INFO_EID, 1482
 CFE_ES_CDS_DUMP_ERR_EID, 1482
 CFE_ES_CDS_NAME_ERR_EID, 1482
 CFE_ES_CDS_OWNER_ACTIVE_EID, 1482
 CFE_ES_CDS_REG_DUMP_INF_EID, 1483
 CFE_ES_CDS_REGISTER_ERR_EID, 1483
 CFE_ES_CREATING_CDS_DUMP_ERR_EID, 1483
 CFE_ES_ERLOG1_INF_EID, 1483
 CFE_ES_ERLOG2_EID, 1484
 CFE_ES_ERLOG2_ERR_EID, 1484
 CFE_ES_ERLOG_PENDING_ERR_EID, 1484
 CFE_ES_ERR_SYSLOGMODE_EID, 1484
 CFE_ES_ERREXIT_APP_ERR_EID, 1485
 CFE_ES_ERREXIT_APP_INF_EID, 1485
 CFE_ES_EXIT_APP_ERR_EID, 1485
 CFE_ES_EXIT_APP_INF_EID, 1485
 CFE_ES_FILEWRITE_ERR_EID, 1486
 CFE_ES_INIT_INF_EID, 1486
 CFE_ES_INITSTATS_INF_EID, 1486
 CFE_ES_INVALID_POOL_HANDLE_ERR_EID,
 1486
 CFE_ES_LEN_ERR_EID, 1487
 CFE_ES_MID_ERR_EID, 1487
 CFE_ES_NOOP_INF_EID, 1487
 CFE_ES_ONE_APP_EID, 1487
 CFE_ES_ONE_APPID_ERR_EID, 1488
 CFE_ES_ONE_ERR_EID, 1488
 CFE_ES_OSCREATE_ERR_EID, 1488
 CFE_ES_PCR_ERR1_EID, 1488
 CFE_ES_PCR_ERR2_EID, 1489
 CFE_ES_PERF_DATAWRITTEN_EID, 1489
 CFE_ES_PERF_FILTMSKCMD_EID, 1489
 CFE_ES_PERF_FILTMSKERR_EID, 1489
 CFE_ES_PERF_LOG_ERR_EID, 1490
 CFE_ES_PERF_STARTCMD_EID, 1490
 CFE_ES_PERF_STARTCMD_ERR_EID, 1490

CFE_ES_PERF_STARTCMD_TRIG_ERR_EID, 1490
 CFE_ES_PERF_STOPCMD_EID, 1491
 CFE_ES_PERF_STOPCMD_ERR2_EID, 1491
 CFE_ES_PERF_TRIGMSKCMD_EID, 1491
 CFE_ES_PERF_TRIGMSKERR_EID, 1491
 CFE_ES_RELOAD_APP_DBG_EID, 1492
 CFE_ES_RELOAD_APP_ERR1_EID, 1492
 CFE_ES_RELOAD_APP_ERR2_EID, 1492
 CFE_ES_RELOAD_APP_ERR3_EID, 1492
 CFE_ES_RELOAD_APP_ERR4_EID, 1493
 CFE_ES_RELOAD_APP_INF_EID, 1493
 CFE_ES_RESET_INF_EID, 1493
 CFE_ES_RESET_PR_COUNT_EID, 1493
 CFE_ES_RESTART_APP_DBG_EID, 1494
 CFE_ES_RESTART_APP_ERR1_EID, 1494
 CFE_ES_RESTART_APP_ERR2_EID, 1494
 CFE_ES_RESTART_APP_ERR3_EID, 1494
 CFE_ES_RESTART_APP_ERR4_EID, 1495
 CFE_ES_RESTART_APP_INF_EID, 1495
 CFE_ES_SET_MAX_PR_COUNT_EID, 1495
 CFE_ES_START_ERR_EID, 1495
 CFE_ES_START_EXC_ACTION_ERR_EID, 1496
 CFE_ES_START_INF_EID, 1496
 CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID, 1496
 CFE_ES_START_INVALID_FILENAME_ERR_EID, 1496
 CFE_ES_START_NULL_APP_NAME_ERR_EID, 1497
 CFE_ES_START_PRIORITY_ERR_EID, 1497
 CFE_ES_STOP_DBG_EID, 1497
 CFE_ES_STOP_ERR1_EID, 1497
 CFE_ES_STOP_ERR2_EID, 1498
 CFE_ES_STOP_ERR3_EID, 1498
 CFE_ES_STOP_INF_EID, 1498
 CFE_ES_SYSLOG1_INF_EID, 1498
 CFE_ES_SYSLOG2_EID, 1499
 CFE_ES_SYSLOG2_ERR_EID, 1499
 CFE_ES_SYSLOGMODE_EID, 1499
 CFE_ES_TASKINFO_EID, 1499
 CFE_ES_TASKINFO_OSCREATE_ERR_EID, 1500
 CFE_ES_TASKINFO_WR_ERR_EID, 1500
 CFE_ES_TASKINFO_WRHDR_ERR_EID, 1500
 CFE_ES_TASKWR_ERR_EID, 1500
 CFE_ES_TLM_POOL_STATS_INFO_EID, 1501
 CFE_ES_VERSION_INF_EID, 1501
 CFE_ES_WRHDR_ERR_EID, 1501
 CFE_ES_WRITE_CFE_HDR_ERR_EID, 1501
 CFE_ES_ExceptionAction
 default_cfe_es_extern_typedefs.h, 1420
 CFE_ES_ExceptionAction_Enum_t
 default_cfe_es_extern_typedefs.h, 1417
 CFE_ES_ExceptionAction_PROC_RESTART
 default_cfe_es_extern_typedefs.h, 1420
 CFE_ES_ExceptionAction_RESTART_APP
 default_cfe_es_extern_typedefs.h, 1420
 CFE_ES_EXIT_APP_ERR_EID
 cfe_es_eventids.h, 1485
 CFE_ES_EXIT_APP_INF_EID
 cfe_es_eventids.h, 1485
 CFE_ES_ExitApp
 cFE Application Behavior APIs, 265
 CFE_ES_ExitChildTask
 cFE Child Task APIs, 280
 CFE_ES_FILE_CLOSE_ERR
 cFE Return Code Defines, 242
 CFE_ES_FILE_IO_ERR
 cFE Return Code Defines, 242
 CFE_ES_FileNameCmd, 671
 CommandHeader, 671
 Payload, 671
 CFE_ES_FileNameCmd_Payload, 672
 FileName, 672
 CFE_ES_FileNameCmd_Payload_t
 default_cfe_es_msgdefs.h, 1468
 CFE_ES_FileNameCmd_t
 default_cfe_es_msgstruct.h, 1473
 CFE_ES_FILEWRITE_ERR_EID
 cfe_es_eventids.h, 1486
 CFE_ES_GetAppID
 cFE Information APIs, 269
 CFE_ES_GetAppIDByName
 cFE Information APIs, 270
 CFE_ES_GetAppInfo
 cFE Information APIs, 270
 CFE_ES_GetAppName
 cFE Information APIs, 271
 CFE_ES_GetCDSBlockIDByName
 cFE Critical Data Store APIs, 286
 CFE_ES_GetCDSBlockName
 cFE Critical Data Store APIs, 286
 CFE_ES_GetGenCount
 cFE Generic Counter APIs, 300
 CFE_ES_GetGenCounterIDByName
 cFE Generic Counter APIs, 300
 CFE_ES_GetGenCounterName
 cFE Generic Counter APIs, 301
 CFE_ES_GetLibIDByName
 cFE Information APIs, 272
 CFE_ES_GetLibInfo
 cFE Information APIs, 272
 CFE_ES_GetLibName
 cFE Information APIs, 273
 CFE_ES_GetMemPoolStats
 cFE Memory Manager APIs, 290
 CFE_ES_GetModuleInfo
 cFE Information APIs, 274

CFE_ES_GetPoolBuf
 cFE Memory Manager APIs, 291

CFE_ES_GetPoolBufInfo
 cFE Memory Manager APIs, 291

CFE_ES_GetResetType
 cFE Information APIs, 275

CFE_ES_GetTaskID
 cFE Information APIs, 275

CFE_ES_GetTaskIDByName
 cFE Child Task APIs, 280

CFE_ES_GetTaskInfo
 cFE Information APIs, 276

CFE_ES_GetTaskName
 cFE Child Task APIs, 281

CFE_ES_HK_TLM_MID
 default_cfe_es_msgids.h, 1471

CFE_ES_HousekeepingTlm, 672
 Payload, 672
 TelemetryHeader, 672

CFE_ES_HousekeepingTlm_Payload, 673
 BootSource, 675
 CFECoreChecksum, 675
 CFEMajorVersion, 675
 CFEMinorVersion, 675
 CFEMissionRevision, 675
 CFERevision, 675
 CommandCounter, 675
 CommandErrorCounter, 676
 ERLogEntries, 676
 ERLogIndex, 676
 HeapBlocksFree, 676
 HeapBytesFree, 676
 HeapMaxBlockSize, 676
 MaxProcessorResets, 676
 OSALMajorVersion, 677
 OSALMinorVersion, 677
 OSALMissionRevision, 677
 OSALRevision, 677
 PerfDataCount, 677
 PerfDataEnd, 677
 PerfDataStart, 677
 PerfDataToWrite, 678
 PerfFilterMask, 678
 PerfMode, 678
 PerfState, 678
 PerfTriggerCount, 678
 PerfTriggerMask, 678
 ProcessorResets, 678
 PSPMajorVersion, 679
 PSPMinorVersion, 679
 PSPMissionRevision, 679
 PSPRevision, 679
 RegisteredCoreApps, 679
 RegisteredExternalApps, 679
 RegisteredLibs, 679
 RegisteredTasks, 680
 ResetSubtype, 680
 ResetType, 680
 SysLogBytesUsed, 680
 SysLogEntries, 680
 SysLogMode, 680
 SysLogSize, 680

CFE_ES_HousekeepingTlm_Payload_t
 default_cfe_es_msgdefs.h, 1468

CFE_ES_HousekeepingTlm_t
 default_cfe_es_msgstruct.h, 1473

CFE_ES_IncrementGenCounter
 cFE Generic Counter APIs, 302

CFE_ES_IncrementTaskCounter
 cFE Application Behavior APIs, 265

CFE_ES_INIT_INF_EID
 cfe_es_eventids.h, 1486

CFE_ES_INITSTATS_INF_EID
 cfe_es_eventids.h, 1486

CFE_ES_INVALID_POOL_HANDLE_ERR_EID
 cfe_es_eventids.h, 1486

CFE_ES_LEN_ERR_EID
 cfe_es_eventids.h, 1487

CFE_ES_LIB_ALREADY_LOADED
 cFE Return Code Defines, 243

CFE_ES_LIBID_BASE
 cFE Resource ID base values, 416

CFE_ES_LIBID_C
 cfe_es_api_typedefs.h, 1373

CFE_ES_LibId_t
 default_cfe_es_extern_typedefs.h, 1417

CFE_ES_LibID_ToIndex
 cFE Resource ID APIs, 258

CFE_ES_LIBID_UNDEFINED
 cfe_es_api_typedefs.h, 1373

CFE_ES_LibraryEntryFuncPtr_t
 cfe_es_api_typedefs.h, 1375

CFE_ES_LogEntryType
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_LogEntryType_APPLICATION
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_LogEntryType_CORE
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_LogEntryType_Enum_t
 default_cfe_es_extern_typedefs.h, 1417

CFE_ES_LogMode
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_LogMode_DISCARD
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_LogMode_Enum_t
 default_cfe_es_extern_typedefs.h, 1417

CFE_ES_LogMode_OVERWRITE
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_Main
 cFE Entry/Exit APIs, 260

CFE_ES_MEMADDRESS_C
 default_cfe_es_extern_typedefs.h, 1415

CFE_ES_MemAddress_t
 default_cfe_es_extern_typedefs.h, 1417

CFE_ES_MEMADDRESS_TO_PTR
 default_cfe_es_extern_typedefs.h, 1415

CFE_ES_MEMHANDLE_C
 cfe_es_api_typedefs.h, 1373

CFE_ES_MemHandle_t
 default_cfe_es_extern_typedefs.h, 1418

CFE_ES_MEMHANDLE_UNDEFINED
 cfe_es_api_typedefs.h, 1373

CFE_ES_MEMOFFSET_C
 default_cfe_es_extern_typedefs.h, 1415

CFE_ES_MemOffset_t
 default_cfe_es_extern_typedefs.h, 1418

CFE_ES_MEMOFFSET_TO_SIZE_T
 default_cfe_es_extern_typedefs.h, 1415

CFE_ES_MEMPOOLBUF_C
 cfe_es_api_typedefs.h, 1373

CFE_ES_MemPoolBuf_t
 cfe_es_api_typedefs.h, 1375

CFE_ES_MemPoolStats, 681
 BlockStats, 681
 CheckErrCtr, 681
 NumBlocksRequested, 681
 NumFreeBytes, 682
 PoolSize, 682

CFE_ES_MemPoolStats_t
 default_cfe_es_extern_typedefs.h, 1418

CFE_ES_MEMSTATS_TLM_MID
 default_cfe_es_msgids.h, 1471

CFE_ES_MemStatsTlm, 682
 Payload, 682
 TelemetryHeader, 682

CFE_ES_MemStatsTlm_t
 default_cfe_es_msgstruct.h, 1473

CFE_ES_MID_ERR_EID
 cfe_es_eventids.h, 1487

CFE_ES_MUT_SEM_DELETE_ERR
 cFE Return Code Defines, 243

CFE_ES_NO_MUTEX
 cfe_es_api_typedefs.h, 1374

CFE_ES_NO_RESOURCE_IDS_AVAILABLE
 cFE Return Code Defines, 243

CFE_ES_NOOP_CC
 default_cfe_es_fcncodes.h, 1425

CFE_ES_NOOP_INF_EID
 cfe_es_eventids.h, 1487

CFE_ES_NoopCmd, 683
 CommandHeader, 683

CFE_ES_NoopCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_NOT_IMPLEMENTED
 cFE Return Code Defines, 243

CFE_ES_ONE_APP_EID
 cfe_es_eventids.h, 1487

CFE_ES_ONE_APPID_ERR_EID
 cfe_es_eventids.h, 1488

CFE_ES_ONE_ERR_EID
 cfe_es_eventids.h, 1488

CFE_ES_OneAppTlm, 683
 Payload, 683
 TelemetryHeader, 683

CFE_ES_OneAppTlm_Payload, 684
 ApplInfo, 684

CFE_ES_OneAppTlm_Payload_t
 default_cfe_es_msgdefs.h, 1468

CFE_ES_OneAppTlm_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_OPERATION_TIMED_OUT
 cFE Return Code Defines, 243

CFE_ES_OSCREATE_ERR_EID
 cfe_es_eventids.h, 1488

CFE_ES_OVER_WRITE_SYS_LOG_CC
 default_cfe_es_fcncodes.h, 1426

CFE_ES_OverWriteSysLogCmd, 684
 CommandHeader, 684
 Payload, 684

CFE_ES_OverWriteSysLogCmd_Payload, 685
 Mode, 685

CFE_ES_OverWriteSysLogCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469

CFE_ES_OverWriteSysLogCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_PCR_ERR1_EID
 cfe_es_eventids.h, 1488

CFE_ES_PCR_ERR2_EID
 cfe_es_eventids.h, 1489

CFE_ES_PERF_DATAWRITTEN_EID
 cfe_es_eventids.h, 1489

CFE_ES_PERF_FILTMSKCMD_EID
 cfe_es_eventids.h, 1489

CFE_ES_PERF_FILTMSKERR_EID
 cfe_es_eventids.h, 1489

CFE_ES_PERF_LOG_ERR_EID
 cfe_es_eventids.h, 1490

CFE_ES_PERF_STARTCMD_EID
 cfe_es_eventids.h, 1490

CFE_ES_PERF_STARTCMD_ERR_EID
 cfe_es_eventids.h, 1490

CFE_ES_PERF_STARTCMD_TRIG_ERR_EID
 cfe_es_eventids.h, 1490

CFE_ES_PERF_STOPCMD_EID
 cfe_es_eventids.h, 1491

CFE_ES_PERF_STOPCMD_ERR2_EID
 cfe_es_eventids.h, 1491

cfe_es_eventids.h, 1491
CFE_ES_PERF_TRIGMSKCMD_EID
 cfe_es_eventids.h, 1491
CFE_ES_PERF_TRIGMSKERR_EID
 cfe_es_eventids.h, 1491
CFE_ES_PerfLogAdd
 cFE Performance Monitor APIs, 298
CFE_ES_PerfLogEntry
 cFE Performance Monitor APIs, 297
CFE_ES_PerfLogExit
 cFE Performance Monitor APIs, 297
CFE_ES_PerfMode
 default_cfe_es_msgdefs.h, 1470
CFE_ES_PerfMode_Enum_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_PerfTrigger_CENTER
 default_cfe_es_msgdefs.h, 1470
CFE_ES_PerfTrigger_END
 default_cfe_es_msgdefs.h, 1470
CFE_ES_PerfTrigger_START
 default_cfe_es_msgdefs.h, 1470
CFE_ES_POOL_BLOCK_INVALID
 cFE Return Code Defines, 243
CFE_ES_PoolAlign, 685
 LongDouble, 686
 LongInt, 686
 Ptr, 686
CFE_ES_PoolAlign_t
 cfe_es_api_typedefs.h, 1375
CFE_ES_PoolCreate
 cFE Memory Manager APIs, 292
CFE_ES_PoolCreateEx
 cFE Memory Manager APIs, 293
CFE_ES_PoolCreateNoSem
 cFE Memory Manager APIs, 294
CFE_ES_PoolDelete
 cFE Memory Manager APIs, 295
CFE_ES_POOLID_BASE
 cFE Resource ID base values, 416
CFE_ES_PoolStatsTlm_Payload, 686
 PoolHandle, 686
 PoolStats, 686
CFE_ES_PoolStatsTlm_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_ProcessAsyncEvent
 cFE Miscellaneous APIs, 283
CFE_ES_PutPoolBuf
 cFE Memory Manager APIs, 296
CFE_ES_QUERY_ALL_CC
 default_cfe_es_fcncodes.h, 1426
CFE_ES_QUERY_ALL_TASKS_CC
 default_cfe_es_fcncodes.h, 1427
CFE_ES_QUERY_ONE_CC
 default_cfe_es_fcncodes.h, 1428
CFE_ES_QueryAllCmd, 687
 CommandHeader, 687
 Payload, 687
CFE_ES_QueryAllCmd_t
 default_cfe_es_msgstruct.h, 1474
CFE_ES_QueryAllTasksCmd, 687
 CommandHeader, 688
 Payload, 688
CFE_ES_QueryAllTasksCmd_t
 default_cfe_es_msgstruct.h, 1474
CFE_ES_QueryOneCmd, 688
 CommandHeader, 688
 Payload, 688
CFE_ES_QueryOneCmd_t
 default_cfe_es_msgstruct.h, 1474
CFE_ES_QUEUE_DELETE_ERR
 cFE Return Code Defines, 243
CFE_ES_RegisterCDS
 cFE Critical Data Store APIs, 287
CFE_ES_RegisterGenCounter
 cFE Generic Counter APIs, 302
CFE_ES_RELOAD_APP_CC
 default_cfe_es_fcncodes.h, 1429
CFE_ES_RELOAD_APP_DBG_EID
 cfe_es_eventids.h, 1492
CFE_ES_RELOAD_APP_ERR1_EID
 cfe_es_eventids.h, 1492
CFE_ES_RELOAD_APP_ERR2_EID
 cfe_es_eventids.h, 1492
CFE_ES_RELOAD_APP_ERR3_EID
 cfe_es_eventids.h, 1492
CFE_ES_RELOAD_APP_ERR4_EID
 cfe_es_eventids.h, 1493
CFE_ES_RELOAD_APP_INF_EID
 cfe_es_eventids.h, 1493
CFE_ES_ReloadApp
 cFE Application Control APIs, 262
CFE_ES_ReloadAppCmd, 688
 CommandHeader, 689
 Payload, 689
CFE_ES_ReloadAppCmd_t
 default_cfe_es_msgstruct.h, 1474
CFE_ES_RESET_COUNTERS_CC
 default_cfe_es_fcncodes.h, 1430
CFE_ES_RESET_INF_EID
 cfe_es_eventids.h, 1493
CFE_ES_RESET_PR_COUNT_CC
 default_cfe_es_fcncodes.h, 1431
CFE_ES_RESET_PR_COUNT_EID
 cfe_es_eventids.h, 1493
CFE_ES_ResetCFE
 cFE Entry/Exit APIs, 260
CFE_ES_ResetCountersCmd, 689
 CommandHeader, 689

CFE_ES_ResetCountersCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_ResetPRCountCmd, 689
 CommandHeader, 690

CFE_ES_ResetPRCountCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_RESTART_APP_CC
 default_cfe_es_fcncodes.h, 1431

CFE_ES_RESTART_APP_DBG_EID
 cfe_es_eventids.h, 1494

CFE_ES_RESTART_APP_ERR1_EID
 cfe_es_eventids.h, 1494

CFE_ES_RESTART_APP_ERR2_EID
 cfe_es_eventids.h, 1494

CFE_ES_RESTART_APP_ERR3_EID
 cfe_es_eventids.h, 1494

CFE_ES_RESTART_APP_ERR4_EID
 cfe_es_eventids.h, 1495

CFE_ES_RESTART_APP_INF_EID
 cfe_es_eventids.h, 1495

CFE_ES_RESTART_CC
 default_cfe_es_fcncodes.h, 1432

CFE_ES_RestartApp
 cFE Application Control APIs, 263

CFE_ES_RestartAppCmd, 690
 CommandHeader, 690
 Payload, 690

CFE_ES_RestartAppCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_RestartCmd, 690
 CommandHeader, 691
 Payload, 691

CFE_ES_RestartCmd_Payload, 691
 RestartType, 691

CFE_ES_RestartCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469

CFE_ES_RestartCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_RestoreFromCDS
 cFE Critical Data Store APIs, 288

CFE_ES_RST_ACCESS_ERR
 cFE Return Code Defines, 243

CFE_ES_RunLoop
 cFE Application Behavior APIs, 266

CFE_ES_RunStatus
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_RunStatus_APP_ERROR
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_APP_EXIT
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_APP_RUN
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_CORE_APP_INIT_ERROR
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_Enum_t
 default_cfe_es_extern_typedefs.h, 1418

CFE_ES_RunStatus_MAX
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_SYS_DELETE
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_SYS_EXCEPTION
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_SYS_RELOAD
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_SYS_RESTART
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_RunStatus_UNDEFINED
 default_cfe_es_extern_typedefs.h, 1420

CFE_ES_SEND_HK_MID
 default_cfe_es_msgids.h, 1471

CFE_ES_SEND_MEM_POOL_STATS_CC
 default_cfe_es_fcncodes.h, 1433

CFE_ES_SendHkCmd, 692
 CommandHeader, 692

CFE_ES_SendHkCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_SendMemPoolStatsCmd, 692
 CommandHeader, 692
 Payload, 692

CFE_ES_SendMemPoolStatsCmd_Payload, 693
 Application, 693
 PoolHandle, 693

CFE_ES_SendMemPoolStatsCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469

CFE_ES_SendMemPoolStatsCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_SET_MAX_PR_COUNT_CC
 default_cfe_es_fcncodes.h, 1434

CFE_ES_SET_MAX_PR_COUNT_EID
 cfe_es_eventids.h, 1495

CFE_ES_SET_PERF_FILTER_MASK_CC
 default_cfe_es_fcncodes.h, 1435

CFE_ES_SET_PERF_TRIGGER_MASK_CC
 default_cfe_es_fcncodes.h, 1436

CFE_ES_SetGenCount
 cFE Generic Counter APIs, 303

CFE_ES_SetMaxPRCountCmd, 693
 CommandHeader, 694
 Payload, 694

CFE_ES_SetMaxPRCountCmd_Payload, 694
 MaxPRCount, 694

CFE_ES_SetMaxPRCountCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469

CFE_ES_SetMaxPRCountCmd_t
 default_cfe_es_msgstruct.h, 1474

CFE_ES_SetPerfFilterMaskCmd, 694

CommandHeader, 695
Payload, 695
CFE_ES_SetPerfFilterMaskCmd_Payload, 695
FilterMask, 695
FilterMaskNum, 695
CFE_ES_SetPerfFilterMaskCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_SetPerfFilterMaskCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_SetPerfTriggerMaskCmd, 696
 CommandHeader, 696
 Payload, 696
CFE_ES_SetPerfTriggerMaskCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_SetPerfTrigMaskCmd_Payload, 696
 TriggerMask, 696
 TriggerMaskNum, 697
CFE_ES_SetPerfTrigMaskCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_StackPointer_t
 cfe_es_api_typedefs.h, 1375
CFE_ES_START_APP_CC
 default_cfe_es_fcncodes.h, 1436
CFE_ES_START_ERR_EID
 cfe_es_eventids.h, 1495
CFE_ES_START_EXC_ACTION_ERR_EID
 cfe_es_eventids.h, 1496
CFE_ES_START_INF_EID
 cfe_es_eventids.h, 1496
CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID
 cfe_es_eventids.h, 1496
CFE_ES_START_INVALID_FILENAME_ERR_EID
 cfe_es_eventids.h, 1496
CFE_ES_START_NULL_APP_NAME_ERR_EID
 cfe_es_eventids.h, 1497
CFE_ES_START_PERF_DATA_CC
 default_cfe_es_fcncodes.h, 1437
CFE_ES_START_PRIORITY_ERR_EID
 cfe_es_eventids.h, 1497
CFE_ES_StartApp, 697
 CommandHeader, 697
 Payload, 697
CFE_ES_StartAppCmd_Payload, 697
 AppEntryPoint, 698
 AppFileName, 698
 Application, 698
 ExceptionAction, 698
 Priority, 698
 StackSize, 698
CFE_ES_StartAppCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_StartAppCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_StartPerfCmd_Payload, 699
 TriggerMode, 699
CFE_ES_StartPerfCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_StartPerfDataCmd, 699
 CommandHeader, 699
 Payload, 700
CFE_ES_StartPerfDataCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_STATIC_POOL_TYPE
 cfe_es_api_typedefs.h, 1374
CFE_ES_StatusToString
 cfe_error.h, 1367
CFE_ES_STOP_APP_CC
 default_cfe_es_fcncodes.h, 1438
CFE_ES_STOP_DBG_EID
 cfe_es_eventids.h, 1497
CFE_ES_STOP_ERR1_EID
 cfe_es_eventids.h, 1497
CFE_ES_STOP_ERR2_EID
 cfe_es_eventids.h, 1498
CFE_ES_STOP_ERR3_EID
 cfe_es_eventids.h, 1498
CFE_ES_STOP_INF_EID
 cfe_es_eventids.h, 1498
CFE_ES_STOP_PERF_DATA_CC
 default_cfe_es_fcncodes.h, 1439
CFE_ES_StopAppCmd, 700
 CommandHeader, 700
 Payload, 700
CFE_ES_StopAppCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_StopPerfCmd_Payload, 700
 DataFileName, 701
CFE_ES_StopPerfCmd_Payload_t
 default_cfe_es_msgdefs.h, 1469
CFE_ES_StopPerfDataCmd, 701
 CommandHeader, 701
 Payload, 701
CFE_ES_StopPerfDataCmd_t
 default_cfe_es_msgstruct.h, 1475
CFE_ES_SYSLOG1_INF_EID
 cfe_es_eventids.h, 1498
CFE_ES_SYSLOG2_EID
 cfe_es_eventids.h, 1499
CFE_ES_SYSLOG2_ERR_EID
 cfe_es_eventids.h, 1499
CFE_ES_SYSLOGMODE_EID
 cfe_es_eventids.h, 1499
CFE_ES_SystemState
 default_cfe_es_extern_typedefs.h, 1421
CFE_ES_SystemState_APPS_INIT
 default_cfe_es_extern_typedefs.h, 1421
CFE_ES_SystemState_CORE_READY
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_CORE_STARTUP
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_EARLY_INIT
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_Enum_t
 default_cfe_es_extern_typedefs.h, 1418

CFE_ES_SystemState_MAX
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_OPERATIONAL
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_SHUTDOWN
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_SystemState_UNDEFINED
 default_cfe_es_extern_typedefs.h, 1421

CFE_ES_TASK_DELETE_ERR
 cFE Return Code Defines, 243

CFE_ES_TASK_STACK_ALLOCATE
 cfe_es_api_typedefs.h, 1374

CFE_ES_TaskEntryFuncPtr_t
 cfe_es_api_typedefs.h, 1375

CFE_ES_TASKID_BASE
 cFE Resource ID base values, 416

CFE_ES_TASKID_C
 cfe_es_api_typedefs.h, 1374

CFE_ES_TaskId_t
 default_cfe_es_extern_typedefs.h, 1419

CFE_ES_TaskID_ToIndex
 cFE Resource ID APIs, 259

CFE_ES_TASKID_UNDEFINED
 cfe_es_api_typedefs.h, 1374

CFE_ES_TaskInfo, 701

- ApplId, 702
- AppName, 702
- ExecutionCounter, 702
- Priority, 702
- Spare, 702
- StackSize, 703
- TaskId, 703
- TaskName, 703

CFE_ES_TASKINFO_EID
 cfe_es_eventids.h, 1499

CFE_ES_TASKINFO_OSCREATE_ERR_EID
 cfe_es_eventids.h, 1500

CFE_ES_TaskInfo_t
 default_cfe_es_extern_typedefs.h, 1419

CFE_ES_TASKINFO_WR_ERR_EID
 cfe_es_eventids.h, 1500

CFE_ES_TASKINFO_WRHDR_ERR_EID
 cfe_es_eventids.h, 1500

CFE_ES_TaskPriority_Atom_t
 default_cfe_es_extern_typedefs.h, 1419

CFE_ES_TASKWR_ERR_EID
 cfe_es_eventids.h, 1500

CFE_ES_TEST_LONG_MASK

 cfe_es.h, 1371

CFE_ES_TIMER_DELETE_ERR
 cFE Return Code Defines, 244

CFE_ES_TLM_POOL_STATS_INFO_EID
 cfe_es_eventids.h, 1501

CFE_ES_USE_MUTEX
 cfe_es_api_typedefs.h, 1374

CFE_ES_VERSION_INF_EID
 cfe_es_eventids.h, 1501

CFE_ES_WaitForStartupSync
 cFE Application Behavior APIs, 267

CFE_ES_WaitForSystemState
 cFE Application Behavior APIs, 267

CFE_ES_WRHDR_ERR_EID
 cfe_es_eventids.h, 1501

CFE_ES_WRITE_CFE_HDR_ERR_EID
 cfe_es_eventids.h, 1501

CFE_ES_WRITE_ER_LOG_CC
 default_cfe_es_fcncodes.h, 1440

CFE_ES_WRITE_SYS_LOG_CC
 default_cfe_es_fcncodes.h, 1441

CFE_ES_WriteERLogCmd, 703

- CommandHeader, 703
- Payload, 703

CFE_ES_WriteERLogCmd_t
 default_cfe_es_msgstruct.h, 1475

CFE_ES_WriteSysLogCmd, 704

- CommandHeader, 704
- Payload, 704

CFE_ES_WriteSysLogCmd_t
 default_cfe_es_msgstruct.h, 1475

CFE_ES_WriteToSysLog
 cFE Miscellaneous APIs, 283

CFE_EVENTS_SERVICE
 cfe_error.h, 1365

cfe_evs.h

- CFE_EVS_Send, 1377
- CFE_EVS_SendCrit, 1377
- CFE_EVS_SendDbg, 1377
- CFE_EVS_SendErr, 1377
- CFE_EVS_SendInfo, 1377

CFE_EVS_ADD_EVENT_FILTER_CC
 default_cfe_evs_fcncodes.h, 1505

CFE_EVS_AddEventFilterCmd, 704

- CommandHeader, 704
- Payload, 704

CFE_EVS_AddEventFilterCmd_t
 default_cfe_evs_msgstruct.h, 1534

CFE_EVS_ADDFILTER_EID
 cfe_evs_eventids.h, 1539

cfe_evs_api_typedefs.h

- CFE_EVS_BinFilter_t, 1379
- CFE_EVS_EVERY_FOURTH_ONE, 1378
- CFE_EVS_EVERY_OTHER_ONE, 1378

CFE_EVS_EVERY_OTHER_TWO, 1379
CFE_EVS_FIRST_16_STOP, 1379
CFE_EVS_FIRST_32_STOP, 1379
CFE_EVS_FIRST_4_STOP, 1379
CFE_EVS_FIRST_64_STOP, 1379
CFE_EVS_FIRST_8_STOP, 1379
CFE_EVS_FIRST_ONE_STOP, 1379
CFE_EVS_FIRST_TWO_STOP, 1379
CFE_EVS_NO_FILTER, 1379
CFE_EVS_APP_FILTER_OVERLOAD
 cFE Return Code Defines, 244
CFE_EVS_APP_ILLEGAL_APP_ID
 cFE Return Code Defines, 244
CFE_EVS_APP_NOT_REGISTERED
 cFE Return Code Defines, 244
CFE_EVS_APP_SQUELCHED
 cFE Return Code Defines, 244
CFE_EVS_AppDataCmd_Payload, 705
 AppNameFilename, 705
CFE_EVS_AppDataCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_AppNameBitMaskCmd_Payload, 705
 AppName, 706
 BitMask, 706
 Spare, 706
CFE_EVS_AppNameBitMaskCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_AppNameCmd_Payload, 706
 AppName, 706
CFE_EVS_AppNameCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_AppNameEventIDCmd_Payload, 706
 AppName, 707
 EventID, 707
CFE_EVS_AppNameEventIDCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_AppNameEventIDMaskCmd_Payload, 707
 AppName, 707
 EventID, 708
 Mask, 708
CFE_EVS_AppNameEventIDMaskCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_AppTlmData, 708
 AppEnableStatus, 708
 AppID, 708
 AppMessageSentCounter, 708
 AppMessageSquelchedCounter, 709
CFE_EVS_AppTlmData_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_BinFilter, 709
 EventID, 709
 Mask, 709
CFE_EVS_BinFilter_t
 cfe_evs_api_typedefs.h, 1379
CFE_EVS_BitMaskCmd_Payload, 709
 BitMask, 710
 Spare, 710
CFE_EVS_BitMaskCmd_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_CLEAR_LOG_CC
 default_cfe_evs_fcncodes.h, 1506
CFE_EVS_ClearLogCmd, 710
 CommandHeader, 710
CFE_EVS_ClearLogCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_CMD_MID
 default_cfe_evs_msgids.h, 1532
CFE_EVS_CRITICAL_BIT
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_DEBUG_BIT
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_DELETE_EVENT_FILTER_CC
 default_cfe_evs_fcncodes.h, 1507
CFE_EVS_DeleteEventFilterCmd, 711
 CommandHeader, 711
 Payload, 711
CFE_EVS_DeleteEventFilterCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_DELFILTER_EID
 cfe_evs_eventids.h, 1539
CFE_EVS_DISABLE_APP_EVENT_TYPE_CC
 default_cfe_evs_fcncodes.h, 1507
CFE_EVS_DISABLE_APP_EVENTS_CC
 default_cfe_evs_fcncodes.h, 1508
CFE_EVS_DISABLE_EVENT_TYPE_CC
 default_cfe_evs_fcncodes.h, 1509
CFE_EVS_DISABLE_PORTS_CC
 default_cfe_evs_fcncodes.h, 1510
CFE_EVS_DisableAppEventsCmd, 711
 CommandHeader, 711
 Payload, 711
CFE_EVS_DisableAppEventsCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_DisableAppEventTypeCmd, 712
 CommandHeader, 712
 Payload, 712
CFE_EVS_DisableAppEventTypeCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_DisableEventTypeCmd, 712
 CommandHeader, 713
 Payload, 713
CFE_EVS_DisableEventTypeCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_DisablePortsCmd, 713
 CommandHeader, 713
 Payload, 713
CFE_EVS_DisablePortsCmd_t
 default_cfe_evs_msgstruct.h, 1534

CFE_EVS_DISAPPENTTYPE_EID
 cfe_evs_eventids.h, 1539
CFE_EVS_DISAPPEVT_EID
 cfe_evs_eventids.h, 1539
CFE_EVS_DISEVTTYPE_EID
 cfe_evs_eventids.h, 1540
CFE_EVS_DISPORT_EID
 cfe_evs_eventids.h, 1540
CFE_EVS_ENAAPPEVT_EID
 cfe_evs_eventids.h, 1540
CFE_EVS_ENAAPPEVTTYPE_EID
 cfe_evs_eventids.h, 1540
CFE_EVS_ENABLE_APP_EVENT_TYPE_CC
 default_cfe_evs_fcncodes.h, 1511
CFE_EVS_ENABLE_APP_EVENTS_CC
 default_cfe_evs_fcncodes.h, 1512
CFE_EVS_ENABLE_EVENT_TYPE_CC
 default_cfe_evs_fcncodes.h, 1513
CFE_EVS_ENABLE_PORTS_CC
 default_cfe_evs_fcncodes.h, 1514
CFE_EVS_EnableAppEventsCmd, 713
 CommandHeader, 714
 Payload, 714
CFE_EVS_EnableAppEventsCmd_t
 default_cfe_evs_msgstruct.h, 1534
CFE_EVS_EnableEventTypeCmd, 714
 CommandHeader, 714
 Payload, 714
CFE_EVS_EnableEventTypeCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_EnableEventTypeCmd, 715
 CommandHeader, 715
 Payload, 715
CFE_EVS_EnableEventTypeCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_EnablePortsCmd, 715
 CommandHeader, 716
 Payload, 716
CFE_EVS_EnablePortsCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_ENAEVTTYPE_EID
 cfe_evs_eventids.h, 1541
CFE_EVS_ENAPORT_EID
 cfe_evs_eventids.h, 1541
CFE_EVS_ERR_APPNOREGS_EID
 cfe_evs_eventids.h, 1541
CFE_EVS_ERR_CC_EID
 cfe_evs_eventids.h, 1541
CFE_EVS_ERR_CRDATFILE_EID
 cfe_evs_eventids.h, 1542
CFE_EVS_ERR_CRlogfile_EID
 cfe_evs_eventids.h, 1542
CFE_EVS_ERR_EVTIDNOREGS_EID
 cfe_evs_eventids.h, 1542
CFE_EVS_ERR_NOAPPIDFOUND_EID
 cfe_evs_eventids.h, 1542
CFE_EVS_ERR_UNREGISTERED_EVS_APP,
 1544
CFE_EVS_ERR_WRDATFILE_EID, 1544
CFE_EVS_ERR_ILLAPPIDRANGE_EID
 cfe_evs_eventids.h, 1542
CFE_EVS_ERR_ILLEGALFMTMOD_EID
 cfe_evs_eventids.h, 1543
CFE_EVS_ERR_INVALID_BITMASK_EID
 cfe_evs_eventids.h, 1543
CFE_EVS_ERR_LOGMODE_EID
 cfe_evs_eventids.h, 1543
CFE_EVS_ERR_MAXREGSFILTER_EID
 cfe_evs_eventids.h, 1543
CFE_EVS_ERR_MSGID_EID
 cfe_evs_eventids.h, 1544
CFE_EVS_ERR_NOAPPIDFOUND_EID
 cfe_evs_eventids.h, 1544
CFE_EVS_ERR_UNREGISTERED_EVS_APP
 cfe_evs_eventids.h, 1544
CFE_EVS_ERR_WRDATFILE_EID
 cfe_evs_eventids.h, 1544
CFE_EVS_ERR_WRLlogfile_EID
 cfe_evs_eventids.h, 1545
CFE_EVS_ERROR_BIT
 default_cfe_evs_msgdefs.h, 1530
CFE_EVS_EventFilter
 default_cfe_evs_extern_typedefs.h, 1503
CFE_EVS_EventFilter_BINARY
 default_cfe_evs_extern_typedefs.h, 1503
CFE_EVS_EventFilter_Enum_t
 default_cfe_evs_extern_typedefs.h, 1502
cfe_evs_eventids.h
 CFE_EVS_ADDFILTER_EID, 1539
 CFE_EVS_DELFILTER_EID, 1539
 CFE_EVS_DISAPPENTTYPE_EID, 1539
 CFE_EVS_DISAPPEVT_EID, 1539
 CFE_EVS_DISEVTTYPE_EID, 1540
 CFE_EVS_DISPORT_EID, 1540
 CFE_EVS_ENAAPPEVT_EID, 1540
 CFE_EVS_ENAAPPEVTTYPE_EID, 1540
 CFE_EVS_ENAEVTTYPE_EID, 1541
 CFE_EVS_ENAPORT_EID, 1541
 CFE_EVS_ERR_APPNOREGS_EID, 1541
 CFE_EVS_ERR_CC_EID, 1541
 CFE_EVS_ERR_CRDATFILE_EID, 1542
 CFE_EVS_ERR_CRlogfile_EID, 1542
 CFE_EVS_ERR_EVTIDNOREGS_EID, 1542
 CFE_EVS_ERR_ILLAPPIDRANGE_EID, 1542
 CFE_EVS_ERR_ILLEGALFMTMOD_EID, 1543
 CFE_EVS_ERR_INVALID_BITMASK_EID, 1543
 CFE_EVS_ERR_LOGMODE_EID, 1543
 CFE_EVS_ERR_MAXREGSFILTER_EID, 1543
 CFE_EVS_ERR_MSGID_EID, 1544
 CFE_EVS_ERR_NOAPPIDFOUND_EID, 1544
 CFE_EVS_ERR_UNREGISTERED_EVS_APP, 1544
 CFE_EVS_ERR_WRDATFILE_EID, 1544

CFE_EVS_ERR_WRLOGFILE_EID, [1545](#)
CFE_EVS_EVT_FILTERED_EID, [1545](#)
CFE_EVS_FILTER_MAX_EID, [1545](#)
CFE_EVS_LEN_ERR_EID, [1545](#)
CFE_EVS_LOGMODE_EID, [1546](#)
CFE_EVS_NOOP_EID, [1546](#)
CFE_EVS_RSTALLFILTER_EID, [1546](#)
CFE_EVS_RSTCNT_EID, [1546](#)
CFE_EVS_RSTEVCNT_EID, [1547](#)
CFE_EVS_RSTFILTER_EID, [1547](#)
CFE_EVS_SETEVTFMTMOD_EID, [1547](#)
CFE_EVS_SETFILTERMSK_EID, [1547](#)
CFE_EVS_SQUELCHED_ERR_EID, [1548](#)
CFE_EVS_STARTUP_EID, [1548](#)
CFE_EVS_WRDAT_EID, [1548](#)
CFE_EVS_WRITE_HEADER_ERR_EID, [1548](#)
CFE_EVS_WRLOG_EID, [1549](#)
CFE_EVS_EventOutput
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventOutput_Enum_t
 default_cfe_evs_extern_typedefs.h, [1503](#)
CFE_EVS_EventOutput_PORT1
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventOutput_PORT2
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventOutput_PORT3
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventOutput_PORT4
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventType
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventType_CRITICAL
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventType_DEBUG
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventType_Enum_t
 default_cfe_evs_extern_typedefs.h, [1503](#)
CFE_EVS_EventType_ERROR
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EventType_INFORMATION
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_EVERY_FOURTH_ONE
 cfe_evs_api_typedefs.h, [1378](#)
CFE_EVS_EVERY_OTHER_ONE
 cfe_evs_api_typedefs.h, [1378](#)
CFE_EVS_EVERY_OTHER_TWO
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_EVT_FILTERED_EID
 cfe_evs_eventids.h, [1545](#)
CFE_EVS_EVT_NOT_REGISTERED
 cFE Return Code Defines, [244](#)
CFE_EVS_FILE_WRITE_ERROR
 cFE Return Code Defines, [244](#)
CFE_EVS_FILTER_MAX_EID
 cfe_evs_extern_typedefs.h, [1504](#)
 cfe_evs_eventids.h, [1545](#)
CFE_EVS_FIRST_16_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_32_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_4_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_64_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_8_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_ONE_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_FIRST_TWO_STOP
 cfe_evs_api_typedefs.h, [1379](#)
CFE_EVS_HK_TLM_MID
 default_cfe_evs_msgids.h, [1532](#)
CFE_EVS_HousekeepingTlm, [716](#)
 Payload, [716](#)
 TelemetryHeader, [716](#)
CFE_EVS_HousekeepingTlm_Payload, [716](#)
 AppData, [717](#)
 CommandCounter, [717](#)
 CommandErrorCounter, [718](#)
 LogEnabled, [718](#)
 LogFullFlag, [718](#)
 LogMode, [718](#)
 LogOverflowCounter, [718](#)
 MessageFormatMode, [718](#)
 MessageSendCounter, [718](#)
 MessageTruncCounter, [719](#)
 OutputPort, [719](#)
 Spare1, [719](#)
 Spare2, [719](#)
 Spare3, [719](#)
 UnregisteredAppCounter, [719](#)
CFE_EVS_HousekeepingTlm_Payload_t
 default_cfe_evs_msgdefs.h, [1531](#)
CFE_EVS_HousekeepingTlm_t
 default_cfe_evs_msgstruct.h, [1535](#)
CFE_EVS_INFORMATION_BIT
 default_cfe_evs_msgdefs.h, [1530](#)
CFE_EVS_INVALID_PARAMETER
 cFE Return Code Defines, [244](#)
CFE_EVS_LEN_ERR_EID
 cfe_evs_eventids.h, [1545](#)
CFE_EVS_LogFileCmd_Payload, [720](#)
 LogFilename, [720](#)
CFE_EVS_LogFileCmd_Payload_t
 default_cfe_evs_msgdefs.h, [1531](#)
CFE_EVS_LogMode
 default_cfe_evs_extern_typedefs.h, [1504](#)
CFE_EVS_LogMode_DISCARD
 default_cfe_evs_extern_typedefs.h, [1504](#)

CFE_EVS_LOGMODE_EID
 cfe_evs_eventids.h, 1546

CFE_EVS_LogMode_Enum_t
 default_cfe_evs_extern_typedefs.h, 1503

CFE_EVS_LogMode_OVERWRITE
 default_cfe_evs_extern_typedefs.h, 1504

CFE_EVS_LONG_EVENT_MSG_MID
 default_cfe_evs_msgids.h, 1532

CFE_EVS_LongEventTlm, 720
 Payload, 720
 TelemetryHeader, 720

CFE_EVS_LongEventTlm_Payload, 721
 Message, 721
 PacketID, 721
 Spare1, 721
 Spare2, 721

CFE_EVS_LongEventTlm_Payload_t
 default_cfe_evs_msgdefs.h, 1531

CFE_EVS_LongEventTlm_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_MsgFormat
 default_cfe_evs_extern_typedefs.h, 1504

CFE_EVS_MsgFormat_Enum_t
 default_cfe_evs_extern_typedefs.h, 1503

CFE_EVS_MsgFormat_LONG
 default_cfe_evs_extern_typedefs.h, 1504

CFE_EVS_MsgFormat_SHORT
 default_cfe_evs_extern_typedefs.h, 1504

CFE_EVS_NO_FILTER
 cfe_evs_api_typedefs.h, 1379

CFE_EVS_NOOP_CC
 default_cfe_evs_fcncodes.h, 1515

CFE_EVS_NOOP_EID
 cfe_evs_eventids.h, 1546

CFE_EVS_NoopCmd, 722
 CommandHeader, 722

CFE_EVS_NoopCmd_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_NOT_IMPLEMENTED
 cFE Return Code Defines, 245

CFE_EVS_PacketID, 722
 AppName, 723
 EventID, 723
 EventType, 723
 ProcessorID, 723
 SpacecraftID, 723

CFE_EVS_PacketID_t
 default_cfe_evs_msgdefs.h, 1531

CFE_EVS_PORT1_BIT
 default_cfe_evs_msgdefs.h, 1530

CFE_EVS_PORT2_BIT
 default_cfe_evs_msgdefs.h, 1530

CFE_EVS_PORT3_BIT
 default_cfe_evs_msgdefs.h, 1530

CFE_EVS_PORT4_BIT
 default_cfe_evs_msgdefs.h, 1530

CFE_EVS_Register
 cFE Registration APIs, 305

CFE_EVS_RESET_ALL_FILTERS_CC
 default_cfe_evs_fcncodes.h, 1515

CFE_EVS_RESET_APP_COUNTER_CC
 default_cfe_evs_fcncodes.h, 1516

CFE_EVS_RESET_AREA_POINTER
 cFE Return Code Defines, 245

CFE_EVS_RESET_COUNTERS_CC
 default_cfe_evs_fcncodes.h, 1517

CFE_EVS_RESET_FILTER_CC
 default_cfe_evs_fcncodes.h, 1518

CFE_EVS_ResetAllFilters
 cFE Reset Event Filter APIs, 312

CFE_EVS_ResetAllFiltersCmd, 723
 CommandHeader, 724
 Payload, 724

CFE_EVS_ResetAllFiltersCmd_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_ResetAppCounterCmd, 724
 CommandHeader, 724
 Payload, 724

CFE_EVS_ResetAppCounterCmd_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_ResetCountersCmd, 724
 CommandHeader, 725

CFE_EVS_ResetCountersCmd_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_ResetFilter
 cFE Reset Event Filter APIs, 312

CFE_EVS_ResetFilterCmd, 725
 CommandHeader, 725
 Payload, 725

CFE_EVS_ResetFilterCmd_t
 default_cfe_evs_msgstruct.h, 1535

CFE_EVS_RSTALLFILTER_EID
 cfe_evs_eventids.h, 1546

CFE_EVS_RSTCNT_EID
 cfe_evs_eventids.h, 1546

CFE_EVS_RSTEVCNT_EID
 cfe_evs_eventids.h, 1547

CFE_EVS_RSTFILTER_EID
 cfe_evs_eventids.h, 1547

CFE_EVS_Send
 cfe_evs.h, 1377

CFE_EVS_SEND_HK_MID
 default_cfe_evs_msgids.h, 1532

CFE_EVS_SendCrit
 cfe_evs.h, 1377

CFE_EVS_SendDbg
 cfe_evs.h, 1377

CFE_EVS_SendErr

cfe_evs.h, 1377
CFE_EVS_SendEvent
 cFE Send Event APIs, 307
CFE_EVS_SendEventWithAppID
 cFE Send Event APIs, 308
CFE_EVS_SendHkCmd, 725
 CommandHeader, 726
CFE_EVS_SendHkCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_SendInfo
 cfe_evs.h, 1377
CFE_EVS_SendTimedEvent
 cFE Send Event APIs, 309
CFE_EVS_SET_EVENT_FORMAT_MODE_CC
 default_cfe_evs_fcncodes.h, 1518
CFE_EVS_SET_FILTER_CC
 default_cfe_evs_fcncodes.h, 1519
CFE_EVS_SET_LOG_MODE_CC
 default_cfe_evs_fcncodes.h, 1520
CFE_EVS_SetEventFormatCode_Payload, 726
 MsgFormat, 726
 Spare, 726
CFE_EVS_SetEventFormatMode_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_SetEventFormatModeCmd, 727
 CommandHeader, 727
 Payload, 727
CFE_EVS_SetEventFormatModeCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_SETEVTFMTMOD_EID
 cfe_evs_eventids.h, 1547
CFE_EVS_SetFilterCmd, 727
 CommandHeader, 727
 Payload, 728
CFE_EVS_SetFilterCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_SETFILTERMSK_EID
 cfe_evs_eventids.h, 1547
CFE_EVS_SetLogMode_Payload, 728
 LogMode, 728
 Spare, 728
CFE_EVS_SetLogMode_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_SetLogModeCmd, 728
 CommandHeader, 729
 Payload, 729
CFE_EVS_SetLogModeCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_SHORT_EVENT_MSG_MID
 default_cfe_evs_msgids.h, 1532
CFE_EVS_ShortEventTlm, 729
 Payload, 729
 TelemetryHeader, 729
CFE_EVS_ShortEventTlm_Payload, 730
 PacketID, 730
CFE_EVS_ShortEventTlm_Payload_t
 default_cfe_evs_msgdefs.h, 1531
CFE_EVS_ShortEventTlm_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_SQUELCHED_ERR_EID
 cfe_evs_eventids.h, 1548
CFE_EVS_STARTUP_EID
 cfe_evs_eventids.h, 1548
CFE_EVS_UNKNOWN_FILTER
 cFE Return Code Defines, 245
CFE_EVS_WRDAT_EID
 cfe_evs_eventids.h, 1548
CFE_EVS_WRITE_APP_DATA_FILE_CC
 default_cfe_evs_fcncodes.h, 1521
CFE_EVS_WRITE_HEADER_ERR_EID
 cfe_evs_eventids.h, 1548
CFE_EVS_WRITE_LOG_DATA_FILE_CC
 default_cfe_evs_fcncodes.h, 1522
CFE_EVS_WriteAppDataFileCmd, 730
 CommandHeader, 730
 Payload, 731
CFE_EVS_WriteAppDataFileCmd_t
 default_cfe_evs_msgstruct.h, 1535
CFE_EVS_WriteLogFileCmd, 731
 CommandHeader, 731
 Payload, 731
CFE_EVS_WriteLogFileCmd_t
 default_cfe_evs_msgstruct.h, 1536
CFE_EVS_WRLOG_EID
 cfe_evs_eventids.h, 1549
CFE_EXECUTIVE_SERVICE
 cfe_error.h, 1365
CFE_FILE_SERVICE
 cfe_error.h, 1365
cfe_fs_api_typedefs.h
 CFE_FS_FileCategory_BINARY_DATA_DUMP, 1382
 CFE_FS_FileCategory_DYNAMIC_MODULE, 1382
 CFE_FS_FileCategory_MAX, 1382
 CFE_FS_FileCategory_SCRIPT, 1382
 CFE_FS_FileCategory_t, 1382
 CFE_FS_FileCategory_TEMP, 1382
 CFE_FS_FileCategory_TEXT_LOG, 1382
 CFE_FS_FileCategory_UNKNOWN, 1382
 CFE_FS_FileWriteEvent_COMPLETE, 1383
 CFE_FS_FileWriteEvent_CREATE_ERROR, 1383
 CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR, 1383
 CFE_FS_FileWriteEvent_MAX, 1383
 CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR, 1383
 CFE_FS_FileWriteEvent_t, 1382
 CFE_FS_FileWriteEvent_UNDEFINED, 1383

CFE_FS_FileWriteGetData_t, 1381
 CFE_FS_FileWriteMetaData_t, 1382
 CFE_FS_FileWriteOnEvent_t, 1382
 CFE_FS_BackgroundFileDumpIsPending
 cFE File Utility APIs, 318
 CFE_FS_BackgroundFileDumpRequest
 cFE File Utility APIs, 319
 CFE_FS_BAD_ARGUMENT
 cFE Return Code Defines, 245
 CFE_FS_ExtractFilenameFromPath
 cFE File Utility APIs, 319
 CFE_FS_FILE_CONTENT_ID
 default_cfe_fs_interface_cfg.h, 1551
 example_mission_cfg.h, 1288
 CFE_FS_FileCategory_BINARY_DATA_DUMP
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_DYNAMIC_MODULE
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_MAX
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_SCRIPT
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_t
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_TEMP
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_TEXT_LOG
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileCategory_UNKNOWN
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileWriteEvent_COMPLETE
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteEvent_CREATE_ERROR
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteEvent_MAX
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteEvent_t
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileWriteEvent_UNDEFINED
 cfe_fs_api_typedefs.h, 1383
 CFE_FS_FileWriteGetData_t
 cfe_fs_api_typedefs.h, 1381
 CFE_FS_FileWriteMetaData_t, 731
 Description, 732
 FileName, 732
 FileSubType, 732
 GetData, 732
 IsPending, 732
 OnEvent, 732
 CFE_FS_FileWriteMetaData_t
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FileWriteOnEvent_t
 cfe_fs_api_typedefs.h, 1382
 CFE_FS_FNAME_TOO_LONG
 cFE Return Code Defines, 245
 CFE_FS_GetDefaultExtension
 cFE File Utility APIs, 320
 CFE_FS_GetDefaultMountPoint
 cFE File Utility APIs, 320
 CFE_FS_HDR_DESC_MAX_LEN
 default_cfe_fs_interface_cfg.h, 1552
 example_mission_cfg.h, 1289
 CFE_FS_Header, 733
 ApplicationID, 733
 ContentType, 733
 Description, 733
 Length, 733
 ProcessorID, 734
 SpacecraftID, 734
 SubType, 734
 TimeSeconds, 734
 TimeSubSeconds, 734
 CFE_FS_Header_t
 default_cfe_fs_filedef.h, 1550
 CFE_FS_InitHeader
 cFE File Header Management APIs, 314
 CFE_FS_INVALID_PATH
 cFE Return Code Defines, 245
 CFE_FS_NOT_IMPLEMENTED
 cFE Return Code Defines, 245
 CFE_FS_ParseInputFileName
 cFE File Utility APIs, 320
 CFE_FS_ParseInputFileNameEx
 cFE File Utility APIs, 321
 CFE_FS_ReadHeader
 cFE File Header Management APIs, 314
 CFE_FS_SetTimestamp
 cFE File Header Management APIs, 315
 CFE_FS_SubType
 default_cfe_fs_filedef.h, 1550
 CFE_FS_SubType_Enum_t
 default_cfe_fs_filedef.h, 1550
 CFE_FS_SubType_ES_CDS_REG
 default_cfe_fs_filedef.h, 1551
 CFE_FS_SubType_ES_ERLOG
 default_cfe_fs_filedef.h, 1550
 CFE_FS_SubType_ES_PERFDATA
 default_cfe_fs_filedef.h, 1551
 CFE_FS_SubType_ES_QUERYALL
 default_cfe_fs_filedef.h, 1550
 CFE_FS_SubType_ES_QUERYALLTASKS
 default_cfe_fs_filedef.h, 1551
 CFE_FS_SubType_ES_SYSLOG
 default_cfe_fs_filedef.h, 1550

CFE_FS_SubType_EVS_APPDATA
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_EVS_EVENTLOG
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_SB_MAPDATA
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_SB_PIPE DATA
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_SB_ROUTEDATA
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_TBL_IMG
 default_cfe_fs_filedef.h, 1551

CFE_FS_SubType_TBL_REG
 default_cfe_fs_filedef.h, 1551

CFE_FS_WriteHeader
 cFE File Header Management APIs, 316

CFE_GENERIC_SERVICE
 cfe_error.h, 1366

CFE_GLOBAL_CMD_MID_BASE
 default_cfe_core_api_base_msgids.h, 1349

CFE_GLOBAL_CMD_TOPICID_TO_MIDV
 default_cfe_core_api_base_msgids.h, 1349

CFE_GLOBAL_TLM_MID_BASE
 default_cfe_core_api_base_msgids.h, 1349

CFE_GLOBAL_TLM_TOPICID_TO_MIDV
 default_cfe_core_api_base_msgids.h, 1349

CFE_LAST_OFFICIAL
 cfe_version.h, 1412

CFE_MAJOR_VERSION
 cfe_version.h, 1412

CFE_MAKE_BIG16
 cfe_endian.h, 1359

CFE_MAKE_BIG32
 cfe_endian.h, 1359

CFE_MINOR_VERSION
 cfe_version.h, 1412

CFE_MISSION_CF_CH0_RX_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_CH0_TX_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_CH1_RX_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_CH1_TX_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_CMD_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_EOT_TLM_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_HK_TLM_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_SEND_HK_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_CF_WAKE_UP_TOPICID
 default_cf_topicids.h, 842

CFE_MISSION_ES_APP_TLM_TOPICID
 default_cfe_es_topicids.h, 1476

CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN
 default_cfe_es_interface_cfg.h, 1443

 example_mission_cfg.h, 1289

CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
 default_cfe_es_interface_cfg.h, 1443

 example_mission_cfg.h, 1289

CFE_MISSION_ES_CMD_TOPICID
 default_cfe_es_topicids.h, 1476

CFE_MISSION_ES_CRC_16
 default_cfe_es_interface_cfg.h, 1443

 example_mission_cfg.h, 1289

CFE_MISSION_ES_CRC_32
 default_cfe_es_interface_cfg.h, 1443

 example_mission_cfg.h, 1289

CFE_MISSION_ES_CRC_8
 default_cfe_es_interface_cfg.h, 1444

 example_mission_cfg.h, 1289

CFE_MISSION_ES_DEFAULT_CRC
 default_cfe_es_interface_cfg.h, 1444

 example_mission_cfg.h, 1290

CFE_MISSION_ES_HK_TLM_TOPICID
 default_cfe_es_topicids.h, 1476

CFE_MISSION_ES_MAIN_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_ES_MAX_APPLICATIONS
 default_cfe_es_interface_cfg.h, 1444

 example_mission_cfg.h, 1290

CFE_MISSION_ES_MEMSTATS_TLM_TOPICID
 default_cfe_es_topicids.h, 1476

CFE_MISSION_ES_PERF_EXIT_BIT
 sample_perfids.h, 1343

CFE_MISSION_ES_PERF_MAX_IDS
 default_cfe_es_interface_cfg.h, 1444

 example_mission_cfg.h, 1290

CFE_MISSION_ES_POOL_MAX_BUCKETS
 default_cfe_es_interface_cfg.h, 1444

 example_mission_cfg.h, 1290

CFE_MISSION_ES_SEND_HK_TOPICID
 default_cfe_es_topicids.h, 1476

CFE_MISSION_EVS_CMD_TOPICID
 default_cfe_evs_topicids.h, 1536

CFE_MISSION_EVS_HK_TLM_TOPICID
 default_cfe_evs_topicids.h, 1536

CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID
 default_cfe_evs_topicids.h, 1537

CFE_MISSION_EVS_MAIN_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
 default_cfe_evs_interface_cfg.h, 1523

 example_mission_cfg.h, 1291

CFE_MISSION_EVS_SEND_HK_TOPICID
 default_cfe_evs_topicids.h, 1537

CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID
 default_cfe_evs_topicids.h, 1537

CFE_MISSION_MAX_API_LEN
 default_cfe_core_api_interface_cfg.h, 1350
 example_mission_cfg.h, 1291

CFE_MISSION_MAX_FILE_LEN
 default_cfe_core_api_interface_cfg.h, 1351
 example_mission_cfg.h, 1291

CFE_MISSION_MAX_NUM_FILES
 default_cfe_core_api_interface_cfg.h, 1351
 example_mission_cfg.h, 1292

CFE_MISSION_MAX_PATH_LEN
 default_cfe_core_api_interface_cfg.h, 1352
 example_mission_cfg.h, 1292

CFE_MISSION_REV
 cfe_version.h, 1412

CFE_MISSION_SB_ALLSUBS_TLM_TOPICID
 default_cfe_sb_topicids.h, 1582

CFE_MISSION_SB_CMD_TOPICID
 default_cfe_sb_topicids.h, 1582

CFE_MISSION_SB_HK_TLM_TOPICID
 default_cfe_sb_topicids.h, 1582

CFE_MISSION_SB_MAIN_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_SB_MAX_PIPES
 default_cfe_sb_interface_cfg.h, 1566
 example_mission_cfg.h, 1293

CFE_MISSION_SB_MAX_SB_MSG_SIZE
 default_cfe_sb_interface_cfg.h, 1566
 example_mission_cfg.h, 1293

CFE_MISSION_SB_MSG_LIM_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_SB_ONESUB_TLM_TOPICID
 default_cfe_sb_topicids.h, 1583

CFE_MISSION_SB_PIPE_OFLOW_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_SB_SEND_HK_TOPICID
 default_cfe_sb_topicids.h, 1583

CFE_MISSION_SB_STATS_TLM_TOPICID
 default_cfe_sb_topicids.h, 1583

CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID
 default_cfe_sb_topicids.h, 1583

CFE_MISSION_TBL_CMD_TOPICID
 default_cfe_tbl_topicids.h, 1627

CFE_MISSION_TBL_HK_TLM_TOPICID
 default_cfe_tbl_topicids.h, 1627

CFE_MISSION_TBL_MAIN_PERF_ID
 sample_perfids.h, 1343

CFE_MISSION_TBL_MAX_FULL_NAME_LEN
 default_cfe_tbl_interface_cfg.h, 1613
 example_mission_cfg.h, 1293

CFE_MISSION_TBL_MAX_NAME_LENGTH
 default_cfe_tbl_interface_cfg.h, 1614
 example_mission_cfg.h, 1294

CFE_MISSION_TBL_REG_TLM_TOPICID
 default_cfe_tbl_topicids.h, 1627

CFE_MISSION_TBL_SEND_HK_TOPICID
 default_cfe_tbl_topicids.h, 1627

CFE_MISSION_TIME_AT_TONE_WAS
 default_cfe_time_interface_cfg.h, 1668
 example_mission_cfg.h, 1294

CFE_MISSION_TIME_AT_TONE_WILL_BE
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_CFG_DEFAULT_TAI
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_CFG_DEFAULT_UTC
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_CFG_FAKE_TONE
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_CMD_TOPICID
 default_cfe_time_topicids.h, 1686

CFE_MISSION_TIME_DATA_CMD_TOPICID
 default_cfe_time_topicids.h, 1686

CFE_MISSION_TIME_DEF_DELAY_SECS
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_DEF_DELAY_SUBS
 default_cfe_time_interface_cfg.h, 1669
 example_mission_cfg.h, 1295

CFE_MISSION_TIME_DEF_LEAPS
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_DEF_MET_SECS
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_DEF_MET_SUBS
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_DEF_STCF_SECS
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_DEF_STCF_SUBS
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_DIAG_TLM_TOPICID
 default_cfe_time_topicids.h, 1686

CFE_MISSION_TIME_EPOCH_DAY
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_EPOCH_HOUR
 default_cfe_time_interface_cfg.h, 1670
 example_mission_cfg.h, 1296

CFE_MISSION_TIME_EPOCH_MICROS
 default_cfe_time_interface_cfg.h, 1670

example_mission_cfg.h, 1296
CFE_MISSION_TIME_EPOCH_MINUTE
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_EPOCH_SECOND
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_EPOCH_YEAR
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_FS_FACTOR
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_HK_TLM_TOPICID
 default_cfe_time_topicids.h, 1686
CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID
 sample_perfids.h, 1343
CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID
 sample_perfids.h, 1343
CFE_MISSION_TIME_MAIN_PERF_ID
 sample_perfids.h, 1344
CFE_MISSION_TIME_MAX_ELAPSED
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_MIN_ELAPSED
 default_cfe_time_interface_cfg.h, 1671
 example_mission_cfg.h, 1297
CFE_MISSION_TIME_ONEHZ_CMD_TOPICID
 default_cfe_time_topicids.h, 1686
CFE_MISSION_TIME_SEND_CMD_TOPICID
 default_cfe_time_topicids.h, 1686
CFE_MISSION_TIME_SEND_HK_TOPICID
 default_cfe_time_topicids.h, 1687
CFE_MISSION_TIME_SEDMET_PERF_ID
 sample_perfids.h, 1344
CFE_MISSION_TIME_TONE1HZISR_PERF_ID
 sample_perfids.h, 1344
CFE_MISSION_TIME_TONE1HZTASK_PERF_ID
 sample_perfids.h, 1344
CFE_MISSION_TIME_TONE_CMD_TOPICID
 default_cfe_time_topicids.h, 1687
cfe_msg_api_typedefs.h
 CFE_MSG_Apld_t, 1387
 CFE_MSG_BAD_ARGUMENT, 1386
 CFE_MSG_Checksum_t, 1387
 CFE_MSG_CommandHeader_t, 1387
 CFE_MSG_EDSVersion_t, 1387
 CFE_MSG_Endian, 1388
 CFE_MSG_Endian_Big, 1388
 CFE_MSG_Endian_Invalid, 1388
 CFE_MSG_Endian_Little, 1388
 CFE_MSG_Endian_t, 1387
 CFE_MSG_FcnCode_t, 1387
 CFE_MSG_HeaderVersion_t, 1387
CFE_MSG_Message_t, 1387
CFE_MSG_NOT_IMPLEMENTED, 1386
CFE_MSG_PlaybackFlag, 1388
CFE_MSG_PlaybackFlag_t, 1387
CFE_MSG_PlayFlag_Invalid, 1389
CFE_MSG_PlayFlag_Original, 1389
CFE_MSG_PlayFlag_Playback, 1389
CFE_MSG_SegFlag_Continue, 1389
CFE_MSG_SegFlag_First, 1389
CFE_MSG_SegFlag_Invalid, 1389
CFE_MSG_SegFlag_Last, 1389
CFE_MSG_SegFlag_Unsegmented, 1389
CFE_MSG_SegmentationFlag, 1389
CFE_MSG_SegmentationFlag_t, 1388
CFE_MSG_SequenceCount_t, 1388
CFE_MSG_Size_t, 1388
CFE_MSG_Subsystem_t, 1388
CFE_MSG_System_t, 1388
CFE_MSG_TelemetryHeader_t, 1388
CFE_MSG_Type, 1389
CFE_MSG_Type_Cmd, 1389
CFE_MSG_Type_Invalid, 1389
CFE_MSG_Type_t, 1388
CFE_MSG_Type_Tlm, 1389
CFE_MSG_WRONG_MSG_TYPE, 1387
CFE_MSG_Apld_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_BAD_ARGUMENT
 cfe_msg_api_typedefs.h, 1386
CFE_MSG_Checksum_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_CommandHeader_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_EDSVersion_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_Endian
 cfe_msg_api_typedefs.h, 1388
CFE_MSG_Endian_Big
 cfe_msg_api_typedefs.h, 1388
CFE_MSG_Endian_Invalid
 cfe_msg_api_typedefs.h, 1388
CFE_MSG_Endian_Little
 cfe_msg_api_typedefs.h, 1388
CFE_MSG_Endian_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_FcnCode_t
 cfe_msg_api_typedefs.h, 1387
CFE_MSG_GenerateChecksum
 cFE Message Secondary Header APIs, 339
CFE_MSG_GetApld
 cFE Message Primary Header APIs, 324
CFE_MSG_GetEDSVersion
 cFE Message Extended Header APIs, 333
CFE_MSG_GetEndian

cFE Message Extended Header APIs, [334](#)
CFE_MSG_GetFcnCode
 cFE Message Secondary Header APIs, [340](#)
CFE_MSG_GetHasSecondaryHeader
 cFE Message Primary Header APIs, [325](#)
CFE_MSG_GetHeaderVersion
 cFE Message Primary Header APIs, [325](#)
CFE_MSG_GetMsgId
 cFE Message Id APIs, [344](#)
CFE_MSG_GetMsgTime
 cFE Message Secondary Header APIs, [340](#)
CFE_MSG_GetNextSequenceCount
 cFE Message Primary Header APIs, [326](#)
CFE_MSG_GetPlaybackFlag
 cFE Message Extended Header APIs, [334](#)
CFE_MSG_GetSegmentationFlag
 cFE Message Primary Header APIs, [326](#)
CFE_MSG_GetSequenceCount
 cFE Message Primary Header APIs, [327](#)
CFE_MSG.GetSize
 cFE Message Primary Header APIs, [327](#)
CFE_MSG_GetSubsystem
 cFE Message Extended Header APIs, [335](#)
CFE_MSG_GetSystem
 cFE Message Extended Header APIs, [335](#)
CFE_MSG.GetType
 cFE Message Primary Header APIs, [328](#)
CFE_MSG.GetTypeFromMsgId
 cFE Message Id APIs, [344](#)
CFE_MSG_HeaderVersion_t
 `cfe_msg_api_typedefs.h, 1387`
CFE_MSG_Init
 cFE Generic Message APIs, [323](#)
CFE_MSG_Message_t
 `cfe_msg_api_typedefs.h, 1387`
CFE_MSG_NOT_IMPLEMENTED
 `cfe_msg_api_typedefs.h, 1386`
CFE_MSG_OriginationAction
 cFE Message Integrity APIs, [346](#)
CFE_MSG_PlaybackFlag
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_PlaybackFlag_t
 `cfe_msg_api_typedefs.h, 1387`
CFE_MSG_PlayFlag_Invalid
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_PlayFlag_Original
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_PlayFlag_Playback
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegFlag_Continue
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegFlag_First
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegFlag_Invalid
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegFlag_Last
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegFlag_Unsegmented
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegmentationFlag
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_SegmentationFlag_t
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_SequenceCount_t
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_SetApId
 cFE Message Primary Header APIs, [328](#)
CFE_MSG_SetEDSVersion
 cFE Message Extended Header APIs, [336](#)
CFE_MSG_SetEndian
 cFE Message Extended Header APIs, [336](#)
CFE_MSG_SetFcnCode
 cFE Message Secondary Header APIs, [341](#)
CFE_MSG_SetHasSecondaryHeader
 cFE Message Primary Header APIs, [329](#)
CFE_MSG_SetHeaderVersion
 cFE Message Primary Header APIs, [329](#)
CFE_MSG_SetMsgId
 cFE Message Id APIs, [345](#)
CFE_MSG_SetMsgTime
 cFE Message Secondary Header APIs, [341](#)
CFE_MSG_SetPlaybackFlag
 cFE Message Extended Header APIs, [337](#)
CFE_MSG_SetSegmentationFlag
 cFE Message Primary Header APIs, [330](#)
CFE_MSG_SetSequenceCount
 cFE Message Primary Header APIs, [330](#)
CFE_MSG_SetSize
 cFE Message Primary Header APIs, [331](#)
CFE_MSG_SetSubsystem
 cFE Message Extended Header APIs, [337](#)
CFE_MSG_SetSystem
 cFE Message Extended Header APIs, [338](#)
CFE_MSG_SetType
 cFE Message Primary Header APIs, [331](#)
CFE_MSG_Size_t
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_SubSystem_t
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_System_t
 `cfe_msg_api_typedefs.h, 1388`
CFE_MSG_Type
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_Type_Cmd
 `cfe_msg_api_typedefs.h, 1389`
CFE_MSG_Type_Invalid

cfe_msg_api_typedefs.h, 1389
CFE_MSG_Type_t
 cfe_msg_api_typedefs.h, 1388
CFE_MSG_Type_Tlm
 cfe_msg_api_typedefs.h, 1389
CFE_MSG_ValidateChecksum
 cFE Message Secondary Header APIs, 342
CFE_MSG_VerificationAction
 cFE Message Integrity APIs, 347
CFE_MSG_WRONG_MSG_TYPE
 cfe_msg_api_typedefs.h, 1387
CFE_PLATFORM_CMD_TOPICID_TO_MIDV
 default_cfe_core_api_base_msgids.h, 1350
CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
 example_platform_cfg.h, 1302
CFE_PLATFORM_ENDIAN
 example_platform_cfg.h, 1302
CFE_PLATFORM_ES_APP_KILL_TIMEOUT
 default_cfe_es_internal_cfg.h, 1447
 example_platform_cfg.h, 1302
CFE_PLATFORM_ES_APP_SCAN_RATE
 default_cfe_es_internal_cfg.h, 1447
 example_platform_cfg.h, 1303
CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1303
CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1303
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04
 default_cfe_es_internal_cfg.h, 1448
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1304
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14
 default_cfe_es_internal_cfg.h, 1449
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15
 default_cfe_es_internal_cfg.h, 1450
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16
 default_cfe_es_internal_cfg.h, 1450
 example_platform_cfg.h, 1305
CFE_PLATFORM_ES_CDS_SIZE
 default_cfe_es_internal_cfg.h, 1450
 example_platform_cfg.h, 1306
CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
 default_cfe_es_internal_cfg.h, 1450
 example_platform_cfg.h, 1306
CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE
 default_cfe_es_internal_cfg.h, 1450
 example_platform_cfg.h, 1306
CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
 default_cfe_es_internal_cfg.h, 1451
 example_platform_cfg.h, 1306
CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME
 default_cfe_es_internal_cfg.h, 1451
 example_platform_cfg.h, 1307
CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE
 default_cfe_es_internal_cfg.h, 1451
 example_platform_cfg.h, 1307
CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE
 default_cfe_es_internal_cfg.h, 1452
 example_platform_cfg.h, 1307
CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
 default_cfe_es_internal_cfg.h, 1452
 example_platform_cfg.h, 1308
CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE
 default_cfe_es_internal_cfg.h, 1452
 example_platform_cfg.h, 1308
CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE
 default_cfe_es_internal_cfg.h, 1453
 example_platform_cfg.h, 1308
CFE_PLATFORM_ES_ER_LOG_ENTRIES

default_cfe_es_internal_cfg.h, 1453
example_platform_cfg.h, 1309

CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE
default_cfe_es_internal_cfg.h, 1453
example_platform_cfg.h, 1309

CFE_PLATFORM_ES_MAX_APPLICATIONS
default_cfe_es_internal_cfg.h, 1454
example_platform_cfg.h, 1309

CFE_PLATFORM_ES_MAX_BLOCK_SIZE
default_cfe_es_internal_cfg.h, 1454
example_platform_cfg.h, 1310

CFE_PLATFORM_ES_MAX_GEN_COUNTERS
default_cfe_es_internal_cfg.h, 1454
example_platform_cfg.h, 1310

CFE_PLATFORM_ES_MAX_LIBRARIES
default_cfe_es_internal_cfg.h, 1454
example_platform_cfg.h, 1310

CFE_PLATFORM_ES_MAX_MEMORY_POOLS
default_cfe_es_internal_cfg.h, 1455
example_platform_cfg.h, 1310

CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS
default_cfe_es_internal_cfg.h, 1455
example_platform_cfg.h, 1311

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
default_cfe_es_internal_cfg.h, 1455
example_platform_cfg.h, 1311

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1311

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09
default_cfe_es_internal_cfg.h, 1456
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11

default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1313

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1313

CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN
default_cfe_es_internal_cfg.h, 1457
example_platform_cfg.h, 1313

CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING
default_cfe_es_internal_cfg.h, 1458
example_platform_cfg.h, 1313

CFE_PLATFORM_ES_NONVOL_STARTUP_FILE
default_cfe_es_internal_cfg.h, 1458
example_platform_cfg.h, 1313

CFE_PLATFORM_ES_OBJECT_TABLE_SIZE
default_cfe_es_internal_cfg.h, 1458
example_platform_cfg.h, 1314

CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY
default_cfe_es_internal_cfg.h, 1458
example_platform_cfg.h, 1314

CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
default_cfe_es_internal_cfg.h, 1459
example_platform_cfg.h, 1314

CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE
default_cfe_es_internal_cfg.h, 1459
example_platform_cfg.h, 1314

CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
default_cfe_es_internal_cfg.h, 1459
example_platform_cfg.h, 1315

CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
default_cfe_es_internal_cfg.h, 1460
example_platform_cfg.h, 1315

CFE_PLATFORM_ES_PERF_FILTMASK_ALL
default_cfe_es_internal_cfg.h, 1460
example_platform_cfg.h, 1315

CFE_PLATFORM_ES_PERF_FILTMASK_INIT
default_cfe_es_internal_cfg.h, 1460
example_platform_cfg.h, 1315

CFE_PLATFORM_ES_PERF_FILTMASK_NONE
default_cfe_es_internal_cfg.h, 1460
example_platform_cfg.h, 1316

CFE_PLATFORM_ES_PERF_TRIGMASK_ALL

default_cfe_es_internal_cfg.h, 1461
example_platform_cfg.h, 1316

CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
default_cfe_es_internal_cfg.h, 1461
example_platform_cfg.h, 1316

CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
default_cfe_es_internal_cfg.h, 1461
example_platform_cfg.h, 1316

CFE_PLATFORM_ES_POOL_MAX_BUCKETS
default_cfe_es_internal_cfg.h, 1461
example_platform_cfg.h, 1317

CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
default_cfe_es_internal_cfg.h, 1462
example_platform_cfg.h, 1317

CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
default_cfe_es_internal_cfg.h, 1462
example_platform_cfg.h, 1317

CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVEDCFE_PLATFORM_EVS_PORT_DEFAULT
default_cfe_es_internal_cfg.h, 1462
example_platform_cfg.h, 1318

CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
default_cfe_es_internal_cfg.h, 1463
example_platform_cfg.h, 1318

CFE_PLATFORM_ES_START_TASK_PRIORITY
default_cfe_es_internal_cfg.h, 1463
example_platform_cfg.h, 1318

CFE_PLATFORM_ES_START_TASK_STACK_SIZE
default_cfe_es_internal_cfg.h, 1463
example_platform_cfg.h, 1319

CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC
default_cfe_es_internal_cfg.h, 1464
example_platform_cfg.h, 1319

CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
default_cfe_es_internal_cfg.h, 1464
example_platform_cfg.h, 1319

CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
default_cfe_es_internal_cfg.h, 1464
example_platform_cfg.h, 1320

CFE_PLATFORM_ES_USER_RESERVED_SIZE
default_cfe_es_internal_cfg.h, 1465
example_platform_cfg.h, 1320

CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
default_cfe_es_internal_cfg.h, 1465
example_platform_cfg.h, 1320

CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC
default_cfe_evs_internal_cfg.h, 1524
example_platform_cfg.h, 1321

CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
default_cfe_evs_internal_cfg.h, 1524
example_platform_cfg.h, 1321

CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
default_cfe_evs_internal_cfg.h, 1525
example_platform_cfg.h, 1321

CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
default_cfe_evs_internal_cfg.h, 1525
example_platform_cfg.h, 1322

CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE
default_cfe_evs_internal_cfg.h, 1525
example_platform_cfg.h, 1322

CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
default_cfe_evs_internal_cfg.h, 1525
example_platform_cfg.h, 1322

CFE_PLATFORM_EVS_LOG_MAX
default_cfe_evs_internal_cfg.h, 1526
example_platform_cfg.h, 1323

CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST
default_cfe_evs_internal_cfg.h, 1526
example_platform_cfg.h, 1323

CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
default_cfe_evs_internal_cfg.h, 1526
example_platform_cfg.h, 1323

CFE_PLATFORM_EVS_PORT_DEFAULT
default_cfe_evs_internal_cfg.h, 1527
example_platform_cfg.h, 1324

CFE_PLATFORM_EVS_START_TASK_PRIORITY
default_cfe_evs_internal_cfg.h, 1527
example_platform_cfg.h, 1324

CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
default_cfe_evs_internal_cfg.h, 1527
example_platform_cfg.h, 1324

CFE_PLATFORM_SB_BUF_MEMORY_BYTES
default_cfe_sb_internal_cfg.h, 1568
example_platform_cfg.h, 1325

CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
default_cfe_sb_internal_cfg.h, 1568
example_platform_cfg.h, 1325

CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
default_cfe_sb_internal_cfg.h, 1569
example_platform_cfg.h, 1325

CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
default_cfe_sb_internal_cfg.h, 1569
example_platform_cfg.h, 1326

CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
default_cfe_sb_internal_cfg.h, 1569
example_platform_cfg.h, 1326

CFE_PLATFORM_SB_FILTER_MASK1
default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1326

CFE_PLATFORM_SB_FILTER_MASK2
default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1326

CFE_PLATFORM_SB_FILTER_MASK3
default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTER_MASK4
default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTER_MASK5
default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1327

default_cfe_sb_internal_cfg.h, 1570
example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTER_MASK6
 default_cfe_sb_internal_cfg.h, 1570
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTER_MASK7
 default_cfe_sb_internal_cfg.h, 1570
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTER_MASK8
 default_cfe_sb_internal_cfg.h, 1570
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTERED_EVENT1
 default_cfe_sb_internal_cfg.h, 1570
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTERED_EVENT2
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTERED_EVENT3
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTERED_EVENT4
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1327

CFE_PLATFORM_SB_FILTERED_EVENT5
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_FILTERED_EVENT6
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_FILTERED_EVENT7
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_FILTERED_EVENT8
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
 default_cfe_sb_internal_cfg.h, 1571
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_MAX_BLOCK_SIZE
 default_cfe_sb_internal_cfg.h, 1572
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_MAX_DEST_PER_PKT
 default_cfe_sb_internal_cfg.h, 1572
 example_platform_cfg.h, 1328

CFE_PLATFORM_SB_MAX_MSG_IDS
 default_cfe_sb_internal_cfg.h, 1572
 example_platform_cfg.h, 1329

CFE_PLATFORM_SB_MAX_PIPES
 default_cfe_sb_internal_cfg.h, 1572
 example_platform_cfg.h, 1329

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1329

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1329

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07
 default_cfe_sb_internal_cfg.h, 1573
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1330

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_START_TASK_PRIORITY
 default_cfe_sb_internal_cfg.h, 1574
 example_platform_cfg.h, 1331

CFE_PLATFORM_SB_START_TASK_STACK_SIZE
 default_cfe_sb_internal_cfg.h, 1575
 example_platform_cfg.h, 1331

CFE_PLATFORM_TBL_BUF_MEMORY_BYTES
 default_cfe_tbl_internal_cfg.h, 1615
 example_platform_cfg.h, 1332

CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE

default_cfe_tbl_internal_cfg.h, 1615
example_platform_cfg.h, 1332

CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
default_cfe_tbl_internal_cfg.h, 1615
example_platform_cfg.h, 1332

CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
default_cfe_tbl_internal_cfg.h, 1616
example_platform_cfg.h, 1333

CFE_PLATFORM_TBL_MAX_NUM_HANDLES
default_cfe_tbl_internal_cfg.h, 1616
example_platform_cfg.h, 1333

CFE_PLATFORM_TBL_MAX_NUM_TABLES
default_cfe_tbl_internal_cfg.h, 1616
example_platform_cfg.h, 1333

CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
default_cfe_tbl_internal_cfg.h, 1616
example_platform_cfg.h, 1334

CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
default_cfe_tbl_internal_cfg.h, 1617
example_platform_cfg.h, 1334

CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
default_cfe_tbl_internal_cfg.h, 1617
example_platform_cfg.h, 1334

CFE_PLATFORM_TBL_START_TASK_PRIORITY
default_cfe_tbl_internal_cfg.h, 1617
example_platform_cfg.h, 1335

CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
default_cfe_tbl_internal_cfg.h, 1618
example_platform_cfg.h, 1335

CFE_PLATFORM_TBL_U32FROM4CHARS
default_cfe_tbl_internal_cfg.h, 1618
example_platform_cfg.h, 1335

CFE_PLATFORM_TBL_VALID_PRID_1
default_cfe_tbl_internal_cfg.h, 1618
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_PRID_2
default_cfe_tbl_internal_cfg.h, 1618
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_PRID_3
default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_PRID_4
default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_PRID_COUNT
default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_SCID_1
default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1336

CFE_PLATFORM_TBL_VALID_SCID_2
default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1337

CFE_PLATFORM_TBL_VALID_SCID_COUNT

default_cfe_tbl_internal_cfg.h, 1619
example_platform_cfg.h, 1337

CFE_PLATFORM_TIME_CFG_CLIENT
default_cfe_time_internal_cfg.h, 1673
example_platform_cfg.h, 1337

CFE_PLATFORM_TIME_CFG_LATCH_FLY
default_cfe_time_internal_cfg.h, 1673
example_platform_cfg.h, 1337

CFE_PLATFORM_TIME_CFG_SERVER
default_cfe_time_internal_cfg.h, 1673
example_platform_cfg.h, 1337

CFE_PLATFORM_TIME_CFG_SIGNAL
default_cfe_time_internal_cfg.h, 1673
example_platform_cfg.h, 1338

CFE_PLATFORM_TIME_CFG_SOURCE
default_cfe_time_internal_cfg.h, 1673
example_platform_cfg.h, 1338

CFE_PLATFORM_TIME_CFG_SRC_GPS
default_cfe_time_internal_cfg.h, 1674
example_platform_cfg.h, 1338

CFE_PLATFORM_TIME_CFG_SRC_MET
default_cfe_time_internal_cfg.h, 1674
example_platform_cfg.h, 1339

CFE_PLATFORM_TIME_CFG_SRC_TIME
default_cfe_time_internal_cfg.h, 1674
example_platform_cfg.h, 1339

CFE_PLATFORM_TIME_CFG_START_FLY
default_cfe_time_internal_cfg.h, 1674
example_platform_cfg.h, 1339

CFE_PLATFORM_TIME_CFG_TONE_LIMIT
default_cfe_time_internal_cfg.h, 1675
example_platform_cfg.h, 1339

CFE_PLATFORM_TIME_CFG_VIRTUAL
default_cfe_time_internal_cfg.h, 1675
example_platform_cfg.h, 1339

CFE_PLATFORM_TIME_MAX_DELTA_SECS
default_cfe_time_internal_cfg.h, 1675
example_platform_cfg.h, 1340

CFE_PLATFORM_TIME_MAX_DELTA_SUBS
default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1340

CFE_PLATFORM_TIME_MAX_LOCAL_SECS
default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1340

CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1341

CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY
default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1341

CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE
default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1341

CFE_PLATFORM_TIME_START_TASK_PRIORITY

default_cfe_time_internal_cfg.h, 1676
example_platform_cfg.h, 1341
CFE_PLATFORM_TIME_START_TASK_STACK_SIZE
default_cfe_time_internal_cfg.h, 1677
example_platform_cfg.h, 1341
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY
default_cfe_time_internal_cfg.h, 1677
example_platform_cfg.h, 1342
CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE
default_cfe_time_internal_cfg.h, 1677
example_platform_cfg.h, 1342
CFE_PLATFORM_TLM_TOPICID_TO_MIDV
default_cfe_core_api_base_msgids.h, 1350
cfe_psp.h
 CFE_PSP_Main, 1739
CFE_PSP_AttachExceptions
 cfe_psp_exception_api.h, 1747
cfe_psp_cache_api.h
 CFE_PSP_FlushCaches, 1739
cfe_psp_cds_api.h
 CFE_PSP_GetCDSSize, 1740
 CFE_PSP_ReadFromCDS, 1740
 CFE_PSP_WriteToCDS, 1740
cfe_psp_eepromaccess_api.h
 CFE_PSP_EepromPowerDown, 1741
 CFE_PSP_EepromPowerUp, 1742
 CFE_PSP_EepromWrite16, 1742
 CFE_PSP_EepromWrite32, 1742
 CFE_PSP_EepromWrite8, 1743
 CFE_PSP_EepromWriteDisable, 1743
 CFE_PSP_EepromWriteEnable, 1743
CFE_PSP_EepromPowerDown
 cfe_psp_eepromaccess_api.h, 1741
CFE_PSP_EepromPowerUp
 cfe_psp_eepromaccess_api.h, 1742
CFE_PSP_EepromWrite16
 cfe_psp_eepromaccess_api.h, 1742
CFE_PSP_EepromWrite32
 cfe_psp_eepromaccess_api.h, 1742
CFE_PSP_EepromWrite8
 cfe_psp_eepromaccess_api.h, 1743
CFE_PSP_EepromWriteDisable
 cfe_psp_eepromaccess_api.h, 1743
CFE_PSP_EepromWriteEnable
 cfe_psp_eepromaccess_api.h, 1743
CFE_PSP_ERROR
 cfe_psp_error.h, 1745
cfe_psp_error.h
 CFE_PSP_ERROR, 1745
 CFE_PSP_ERROR_ADDRESS_MISALIGNED, 1745
 CFE_PSP_ERROR_NOT_IMPLEMENTED, 1745
 CFE_PSP_ERROR_TIMEOUT, 1745
 CFE_PSP_INVALID_INT_NUM, 1745
CFE_PSP_INVALID_MEM_ADDR, 1745
CFE_PSP_INVALID_MEM_ATTR, 1745
CFE_PSP_INVALID_MEM_RANGE, 1745
CFE_PSP_INVALID_MEM_SIZE, 1745
CFE_PSP_INVALID_MEM_TYPE, 1745
CFE_PSP_INVALID_MEM_WORDSIZE, 1745
CFE_PSP_INVALID_MODULE_ID, 1745
CFE_PSP_INVALID_MODULE_NAME, 1745
CFE_PSP_INVALID_POINTER, 1746
CFE_PSP_NO_EXCEPTION_DATA, 1746
CFE_PSP_STATUS_C, 1746
CFE_PSP_STATUS_STRING_LENGTH, 1746
CFE_PSP_Status_t, 1746
CFE_PSP_StatusString_t, 1746
CFE_PSP_StatusToString, 1746
CFE_PSP_SUCCESS, 1746
CFE_PSP_ERROR_ADDRESS_MISALIGNED
 cfe_psp_error.h, 1745
CFE_PSP_ERROR_NOT_IMPLEMENTED
 cfe_psp_error.h, 1745
CFE_PSP_ERROR_TIMEOUT
 cfe_psp_error.h, 1745
cfe_psp_exception_api.h
 CFE_PSP_AttachExceptions, 1747
 CFE_PSP_Exception_CopyContext, 1747
 CFE_PSP_Exception_GetCount, 1748
 CFE_PSP_Exception_GetSummary, 1748
 CFE_PSP_SetDefaultExceptionEnvironment, 1748
CFE_PSP_Exception_CopyContext
 cfe_psp_exception_api.h, 1747
CFE_PSP_Exception_GetCount
 cfe_psp_exception_api.h, 1748
CFE_PSP_Exception_GetSummary
 cfe_psp_exception_api.h, 1748
CFE_PSP_FlushCaches
 cfe_psp_cache_api.h, 1739
CFE_PSP_Get_Timebase
 cfe_psp_timertick_api.h, 1763
CFE_PSP_GetBuildNumber
 cfe_psp_version_api.h, 1765
CFE_PSP_GetCDSSize
 cfe_psp_cds_api.h, 1740
CFE_PSP_GetCFETextSegmentInfo
 cfe_psp_memrange_api.h, 1755
CFE_PSP_GetKernelTextSegmentInfo
 cfe_psp_memrange_api.h, 1755
CFE_PSP_GetProcessorId
 cfe_psp_id_api.h, 1749
CFE_PSP_GetProcessorName
 cfe_psp_id_api.h, 1749
CFE_PSP_GetResetArea
 cfe_psp_memrange_api.h, 1756
CFE_PSP_GetRestartType
 cfe_psp_watchdog_api.h, 1769

CFE_PSP_GetSpacecraftId
 cfe_psp_id_api.h, 1749

CFE_PSP_GetTime
 cfe_psp_timertick_api.h, 1763

CFE_PSP_GetTimerLow32Rollover
 cfe_psp_timertick_api.h, 1764

CFE_PSP_GetTimerTicksPerSecond
 cfe_psp_timertick_api.h, 1764

CFE_PSP.GetUserReservedArea
 cfe_psp_memrange_api.h, 1756

CFE_PSP_GetVersionCodeName
 cfe_psp_version_api.h, 1765

CFE_PSP_GetVersionNumber
 cfe_psp_version_api.h, 1765

CFE_PSP_GetVersionString
 cfe_psp_version_api.h, 1765

CFE_PSP_GetVolatileDiskMem
 cfe_psp_memrange_api.h, 1756

cfe_psp_id_api.h
 CFE_PSP_GetProcessorId, 1749
 CFE_PSP_GetProcessorName, 1749
 CFE_PSP_GetSpacecraftId, 1749

CFE_PSP_InitSSR
 cfe_psp_ssr_api.h, 1762

CFE_PSP_INVALID_INT_NUM
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_ADDR
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_ATTR
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_RANGE
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_SIZE
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_TYPE
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MEM_WORDSIZE
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MODULE_ID
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_MODULE_NAME
 cfe_psp_error.h, 1745

CFE_PSP_INVALID_POINTER
 cfe_psp_error.h, 1746

CFE_PSP_Main
 cfe_psp.h, 1739

CFE_PSP_MEM_ANY
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_ATTR_READ
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_ATTR_READWRITE
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_ATTR_WRITE
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_EEPROM
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_INVALID
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_RAM
 cfe_psp_memrange_api.h, 1754

CFE_PSP_MEM_SIZE_BYTE
 cfe_psp_memrange_api.h, 1755

CFE_PSP_MEM_SIZE_DWORD
 cfe_psp_memrange_api.h, 1755

CFE_PSP_MEM_SIZE_WORD
 cfe_psp_memrange_api.h, 1755

cfe_psp_memaccess_api.h
 CFE_PSP_MemCpy, 1750
 CFE_PSP_MemRead16, 1750
 CFE_PSP_MemRead32, 1751
 CFE_PSP_MemRead8, 1751
 CFE_PSP_MemSet, 1752
 CFE_PSP_MemWrite16, 1752
 CFE_PSP_MemWrite32, 1752
 CFE_PSP_MemWrite8, 1753

CFE_PSP_MemCpy
 cfe_psp_memaccess_api.h, 1750

cfe_psp_memrange_api.h
 CFE_PSP_GetCFETextSegmentInfo, 1755
 CFE_PSP_GetKernelTextSegmentInfo, 1755
 CFE_PSP_GetResetArea, 1756
 CFE_PSP GetUserReservedArea, 1756
 CFE_PSP_GetVolatileDiskMem, 1756
 CFE_PSP_MEM_ANY, 1754
 CFE_PSP_MEM_ATTR_READ, 1754
 CFE_PSP_MEM_ATTR_READWRITE, 1754
 CFE_PSP_MEM_ATTR_WRITE, 1754
 CFE_PSP_MEM_EEPROM, 1754
 CFE_PSP_MEM_INVALID, 1754
 CFE_PSP_MEM_RAM, 1754
 CFE_PSP_MEM_SIZE_BYTE, 1755
 CFE_PSP_MEM_SIZE_DWORD, 1755
 CFE_PSP_MEM_SIZE_WORD, 1755
 CFE_PSP_MemRangeGet, 1756
 CFE_PSP_MemRanges, 1757
 CFE_PSP_MemRangeSet, 1757
 CFE_PSP_MemValidateRange, 1758

CFE_PSP_MemRangeGet
 cfe_psp_memrange_api.h, 1756

CFE_PSP_MemRanges
 cfe_psp_memrange_api.h, 1757

CFE_PSP_MemRangeSet
 cfe_psp_memrange_api.h, 1757

CFE_PSP_MemRead16
 cfe_psp_memaccess_api.h, 1750

CFE_PSP_MemRead32
 cfe_psp_memaccess_api.h, 1751

CFE_PSP_MemRead8

cfe_psp_memaccess_api.h, 1751
CFE_PSP_MemSet
 cfe_psp_memaccess_api.h, 1752
CFE_PSP_MemValidateRange
 cfe_psp_memrange_api.h, 1758
CFE_PSP_MemWrite16
 cfe_psp_memaccess_api.h, 1752
CFE_PSP_MemWrite32
 cfe_psp_memaccess_api.h, 1752
CFE_PSP_MemWrite8
 cfe_psp_memaccess_api.h, 1753
CFE_PSP_NO_EXCEPTION_DATA
 cfe_psp_error.h, 1746
CFE_PSP_Panic
 cfe_psp_watchdog_api.h, 1769
CFE_PSP_PANIC_CORE_APP
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_GENERAL_FAILURE
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_MEMORY_ALLOC
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_NONVOL_DISK
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_STARTUP
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_STARTUP_SEM
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_PANIC_VOLATILE_DISK
 cfe_psp_watchdog_api.h, 1767
cfe_psp_port_api.h
 CFE_PSP_PortRead16, 1759
 CFE_PSP_PortRead32, 1760
 CFE_PSP_PortRead8, 1760
 CFE_PSP_PortWrite16, 1760
 CFE_PSP_PortWrite32, 1761
 CFE_PSP_PortWrite8, 1761
CFE_PSP_PortRead16
 cfe_psp_port_api.h, 1759
CFE_PSP_PortRead32
 cfe_psp_port_api.h, 1760
CFE_PSP_PortRead8
 cfe_psp_port_api.h, 1760
CFE_PSP_PortWrite16
 cfe_psp_port_api.h, 1760
CFE_PSP_PortWrite32
 cfe_psp_port_api.h, 1761
CFE_PSP_PortWrite8
 cfe_psp_port_api.h, 1761
CFE_PSP_ReadFromCDS
 cfe_psp_cds_api.h, 1740
CFE_PSP_Restart
 cfe_psp_watchdog_api.h, 1769
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_RST_SUBTYPE_EXCEPTION
 cfe_psp_watchdog_api.h, 1767
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_MAX
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_POWER_CYCLE
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_RESET_COMMAND
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_TYPE_MAX
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_TYPE_POWERON
 cfe_psp_watchdog_api.h, 1768
CFE_PSP_RST_TYPE_PROCESSOR
 cfe_psp_watchdog_api.h, 1769
CFE_PSP_SetDefaultExceptionEnvironment
 cfe_psp_exception_api.h, 1748
CFE_PSP_SOFT_TIMEBASE_NAME
 cfe_psp_timertick_api.h, 1763
cfe_psp_ssr_api.h
 CFE_PSP_InitSSR, 1762
CFE_PSP_STATUS_C
 cfe_psp_error.h, 1746
CFE_PSP_STATUS_STRING_LENGTH
 cfe_psp_error.h, 1746
CFE_PSP_Status_t
 cfe_psp_error.h, 1746
CFE_PSP_StatusString_t
 cfe_psp_error.h, 1746
CFE_PSP_StatusToString
 cfe_psp_error.h, 1746
CFE_PSP_SUCCESS
 cfe_psp_error.h, 1746
cfe_psp_timertick_api.h
 CFE_PSP_Get_Timebase, 1763
 CFE_PSP_GetTime, 1763
 CFE_PSP_GetTimerLow32Rollover, 1764
 CFE_PSP_GetTimerTicksPerSecond, 1764
 CFE_PSP_SOFT_TIMEBASE_NAME, 1763
cfe_psp_version_api.h
 CFE_PSP_GetBuildNumber, 1765
 CFE_PSP_GetVersionCodeName, 1765
 CFE_PSP_GetVersionNumber, 1765
 CFE_PSP_GetVersionString, 1765
 cfe_psp_watchdog_api.h

CFE_PSP_GetRestartType, 1769
CFE_PSP_Panic, 1769
CFE_PSP_PANIC_CORE_APP, 1767
CFE_PSP_PANIC_GENERAL_FAILURE, 1767
CFE_PSP_PANIC_MEMORY_ALLOC, 1767
CFE_PSP_PANIC_NONVOL_DISK, 1767
CFE_PSP_PANIC_STARTUP, 1767
CFE_PSP_PANIC_STARTUP_SEM, 1767
CFE_PSP_PANIC_VOLATILE_DISK, 1767
CFE_PSP_Restart, 1769
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET, 1767
CFE_PSP_RST_SUBTYPE_EXCEPTION, 1767
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND, 1768
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG, 1768
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET, 1768
CFE_PSP_RST_SUBTYPE_MAX, 1768
CFE_PSP_RST_SUBTYPE_POWER_CYCLE, 1768
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON, 1768
CFE_PSP_RST_SUBTYPE_RESET_COMMAND, 1768
CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET, 1768
CFE_PSP_RST_TYPE_MAX, 1768
CFE_PSP_RST_TYPE_POWERON, 1768
CFE_PSP_RST_TYPE_PROCESSOR, 1769
CFE_PSP_WatchdogDisable, 1770
CFE_PSP_WatchdogEnable, 1770
CFE_PSP_WatchdogGet, 1770
CFE_PSP_WatchdogInit, 1770
CFE_PSP_WatchdogService, 1770
CFE_PSP_WatchdogSet, 1770
CFE_PSP_WatchdogDisable
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WatchdogEnable
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WatchdogGet
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WatchdogInit
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WatchdogService
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WatchdogSet
 cfe_psp_watchdog_api.h, 1770
CFE_PSP_WriteToCDS
 cfe_psp_cds_api.h, 1740
cfe_resourceld.h
 CFE_Resourceld_Equal, 1391
 CFE_Resourceld_FindNext, 1391
 CFE_Resourceld_FromInteger, 1393
 CFE_Resourceld_GetBase, 1393
 CFE_Resourceld_GetSerial, 1394
 CFE_Resourceld_IsDefined, 1394
 CFE_RESOURCEID_TEST_DEFINED, 1390
 CFE_RESOURCEID_TEST_EQUAL, 1390
 CFE_RESOURCEID_TO ULONG, 1391
 CFE_Resourceld_ToIndex, 1394
 CFE_Resourceld_ToInteger, 1395
 cfe_resourceld_api_typedefs.h
 CFE_RESOURCEID_RESERVED, 1396
 CFE_RESOURCEID_UNDEFINED, 1396
 cfe_resourceld_basevalue.h
 CFE_RESOURCEID_MAKE_BASE, 1554
 CFE_RESOURCEID_MAX, 1554
 CFE_RESOURCEID_SHIFT, 1554
 CFE_RESOURCEID_CONFIGID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_Resourceld_Equal
 cfe_resourceld.h, 1391
 CFE_RESOURCEID_ES_APPID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_ES_LIBID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_ES_POOLID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_ES_TASKID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_Resourceld_FindNext
 cfe_resourceld.h, 1391
 CFE_Resourceld_FromInteger
 cfe_resourceld.h, 1393
 CFE_Resourceld_GetBase
 cfe_resourceld.h, 1393
 CFE_Resourceld_GetSerial
 cfe_resourceld.h, 1394
 CFE_Resourceld_IsDefined
 cfe_resourceld.h, 1394
 CFE_RESOURCEID_MAKE_BASE
 cfe_resourceld_basevalue.h, 1554
 CFE_RESOURCEID_MAX
 cfe_resourceld_basevalue.h, 1554
 CFE_RESOURCEID_RESERVED
 cfe_resourceld_api_typedefs.h, 1396
 CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_SHIFT
 cfe_resourceld_basevalue.h, 1554
 CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET
 cFE Resource ID base values, 415
 CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET
 cFE Resource ID base values, 415

CFE_RESOURCEID_TEST_DEFINED
 cfe_resourceid.h, 1390

CFE_RESOURCEID_TEST_EQUAL
 cfe_resourceid.h, 1390

CFE_RESOURCEID_TO ULONG
 cfe_resourceid.h, 1391

CFE_Resourceld_ToIndex
 cfe_resourceid.h, 1394

CFE_Resourceld_ToInteger
 cfe_resourceid.h, 1395

CFE_RESOURCEID_UNDEFINED
 cfe_resourceid_api_typedefs.h, 1396

CFE_REVISION
 cfe_version.h, 1412

cfe_sb.h
 CFE_BIT, 1398
 CFE_CLR, 1399
 CFE_SET, 1399
 CFE_TST, 1399

CFE_SB_AllocateMessageBuffer
 cFE Zero Copy APIs, 361

CFE_SB_ALLSUBS_TLM_MID
 default_cfe_sb_msgids.h, 1579

CFE_SB_AllSubscriptionsTlm, 734
 Payload, 734
 TelemetryHeader, 735

CFE_SB_AllSubscriptionsTlm_Payload, 735
 Entries, 735
 Entry, 735
 PktSegment, 735
 TotalSegments, 736

CFE_SB_AllSubscriptionsTlm_Payload_t
 default_cfe_sb_msgdefs.h, 1577

CFE_SB_AllSubscriptionsTlm_t
 default_cfe_sb_msgstruct.h, 1580

cfe_sb_api_typedefs.h
 CFE_SB_Buffer_t, 1402
 CFE_SB_DEFAULT_QOS, 1400
 CFE_SB_INVALID_MSG_ID, 1400
 CFE_SB_INVALID_PIPE, 1400
 CFE_SB_MSGID_C, 1401
 CFE_SB_MSGID_RESERVED, 1401
 CFE_SB_MSGID_UNWRAP_VALUE, 1401
 CFE_SB_MSGID_WRAP_VALUE, 1401
 CFE_SB_PEND_FOREVER, 1401
 CFE_SB PIPEID_C, 1402
 CFE_SB_POLL, 1402
 CFE_SB_SUBSCRIPTION, 1402
 CFE_SB_UNSUBSCRIPTION, 1402

CFE_SB_BAD_ARGUMENT
 cFE Return Code Defines, 245

CFE_SB_BAD_CMD_CODE_EID
 cfe_sb_eventids.h, 1586

CFE_SB_BAD_MSGID_EID
 cfe_sb_eventids.h, 1586

CFE_SB_BAD_PIPEID_EID
 cfe_sb_eventids.h, 1586

CFE_SB_BUF_ALOC_ERR
 cFE Return Code Defines, 245

CFE_SB_BUFFER_INVALID
 cFE Return Code Defines, 246

CFE_SB_Buffer_t
 cfe_sb_api_typedefs.h, 1402

CFE_SB_CMD0_RCVD_EID
 cfe_sb_eventids.h, 1586

CFE_SB_CMD1_RCVD_EID
 cfe_sb_eventids.h, 1587

CFE_SB_CMD_MID
 default_cfe_sb_msgids.h, 1579

CFE_SB_CmdTopicIdToMsgId
 cFE Message ID APIs, 369

CFE_SB_CR_PIPE_BAD_ARG_EID
 cfe_sb_eventids.h, 1587

CFE_SB_CR_PIPE_ERR_EID
 cfe_sb_eventids.h, 1587

CFE_SB_CR_PIPE_NAME_TAKEN_EID
 cfe_sb_eventids.h, 1587

CFE_SB_CR_PIPE_NO_FREE_EID
 cfe_sb_eventids.h, 1588

CFE_SB_CreatePipe
 cFE Pipe Management APIs, 348

CFE_SB_DEFAULT_QOS
 cfe_sb_api_typedefs.h, 1400

CFE_SB_DEL_PIPE_ERR1_EID
 cfe_sb_eventids.h, 1588

CFE_SB_DEL_PIPE_ERR2_EID
 cfe_sb_eventids.h, 1588

CFE_SB_DeletePipe
 cFE Pipe Management APIs, 349

CFE_SB_DEST_BLK_ERR_EID
 cfe_sb_eventids.h, 1588

CFE_SB_DISABLE_ROUTE_CC
 default_cfe_sb_fcncodes.h, 1557

CFE_SB_DISABLE_SUB_REPORTING_CC
 default_cfe_sb_fcncodes.h, 1558

CFE_SB_DisableRouteCmd, 736
 CommandHeader, 736
 Payload, 736

CFE_SB_DisableRouteCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_DisableSubReportingCmd, 736
 CommandHeader, 737

CFE_SB_DisableSubReportingCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_DSBL_RTE1_EID
 cfe_sb_eventids.h, 1589

CFE_SB_DSBL_RTE2_EID
 cfe_sb_eventids.h, 1589

CFE_SB_DSBL RTE3_EID
cfe_sb_eventids.h, 1589

CFE_SB_DUP_SUBSCRIPT_EID
cfe_sb_eventids.h, 1589

CFE_SB_ENABLE_ROUTE_CC
default_cfe_sb_fcncodes.h, 1558

CFE_SB_ENABLE_SUB_REPORTING_CC
default_cfe_sb_fcncodes.h, 1559

CFE_SB_EnableRouteCmd, 737
CommandHeader, 737
Payload, 737

CFE_SB_EnableRouteCmd_t
default_cfe_sb_msgstruct.h, 1581

CFE_SB_EnableSubReportingCmd, 737
CommandHeader, 738

CFE_SB_EnableSubReportingCmd_t
default_cfe_sb_msgstruct.h, 1581

CFE_SB_ENBL RTE1_EID
cfe_sb_eventids.h, 1590

CFE_SB_ENBL RTE2_EID
cfe_sb_eventids.h, 1590

CFE_SB_ENBL RTE3_EID
cfe_sb_eventids.h, 1590

cfe_sb_eventids.h
CFE_SB_BAD_CMD_CODE_EID, 1586
CFE_SB_BAD_MSGID_EID, 1586
CFE_SB_BAD_PIPEID_EID, 1586
CFE_SB_CMD0_RCVD_EID, 1586
CFE_SB_CMD1_RCVD_EID, 1587
CFE_SB_CR_PIPE_BAD_ARG_EID, 1587
CFE_SB_CR_PIPE_ERR_EID, 1587
CFE_SB_CR_PIPE_NAME_TAKEN_EID, 1587
CFE_SB_CR_PIPE_NO_FREE_EID, 1588
CFE_SB_DEL_PIPE_ERR1_EID, 1588
CFE_SB_DEL_PIPE_ERR2_EID, 1588
CFE_SB_DEST_BLK_ERR_EID, 1588
CFE_SB_DSBL RTE1_EID, 1589
CFE_SB_DSBL RTE2_EID, 1589
CFE_SB_DSBL RTE3_EID, 1589
CFE_SB_DUP_SUBSCRIPT_EID, 1589

CFE_SB_ENBL RTE1_EID, 1590
CFE_SB_ENBL RTE2_EID, 1590
CFE_SB_ENBL RTE3_EID, 1590
CFE_SB_FILEWRITE_ERR_EID, 1590
CFE_SB_FULL_SUB_PKT_EID, 1591
CFE_SB_GET_BUF_ERR_EID, 1591
CFE_SB_GETPIPEIDBYNAME_EID, 1591
CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID, 1591
CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID, 1592
CFE_SB_GETPIPEENAME_EID, 1592
CFE_SB_GETPIPEENAME_ID_ERR_EID, 1592
CFE_SB_GETPIPEENAME_NULL_PTR_EID, 1592

CFE_SB_GETPIPEOPTS_EID, 1593
CFE_SB_GETPIPEOPTS_ID_ERR_EID, 1593
CFE_SB_GETPIPEOPTS_PTR_ERR_EID, 1593
CFE_SB_HASHCOLLISION_EID, 1593
CFE_SB_INIT_EID, 1594
CFE_SB_LEN_ERR_EID, 1594
CFE_SB_MAX_DESTS_MET_EID, 1594
CFE_SB_MAX_MSGS_MET_EID, 1594
CFE_SB_MAX_PIPES_MET_EID, 1595
CFE_SB_MSG_TOO_BIG_EID, 1595
CFE_SB_MSGID_LIM_ERR_EID, 1595
CFE_SB_PART_SUB_PKT_EID, 1595
CFE_SB_PIPE_ADDED_EID, 1596
CFE_SB_PIPE_DELETED_EID, 1596
CFE_SB_Q_FULL_ERR_EID, 1596
CFE_SB_Q_RD_ERR_EID, 1596
CFE_SB_Q_WR_ERR_EID, 1597
CFE_SB_RCV_BAD_ARG_EID, 1597
CFE_SB_RCV_MESSAGE_INTEGRITY_FAIL_EID, 1597
CFE_SB_SEND_BAD_ARG_EID, 1597
CFE_SB_SEND_INV_MSGID_EID, 1598
CFE_SB_SEND_MESSAGE_INTEGRITY_FAIL_EID, 1598
CFE_SB_SEND_NO_SUBS_EID, 1598
CFE_SB_SETPipeOPTS_EID, 1598
CFE_SB_SETPipeOPTS_ID_ERR_EID, 1599
CFE_SB_SETPipeOPTS_OWNER_ERR_EID, 1599
CFE_SB SND RTG_EID, 1599
CFE_SB SND RTG_ERR1_EID, 1599
CFE_SB SND STATS_EID, 1600
CFE_SB SUB ARG_ERR_EID, 1600
CFE_SB SUB INV CALLER_EID, 1600
CFE_SB SUB INV PIPE_EID, 1600
CFE_SB SUBSCRIPTION_RCVD_EID, 1601
CFE_SB SUBSCRIPTION_REMOVED_EID, 1601
CFE_SB SUBSCRIPTION_RPT_EID, 1601
CFE_SB UNSUB ARG_ERR_EID, 1601
CFE_SB UNSUB INV CALLER_EID, 1602
CFE_SB UNSUB INV PIPE_EID, 1602
CFE_SB UNSUB NO SUBS_EID, 1602

CFE_SB_FILEWRITE_ERR_EID
cfe_sb_eventids.h, 1590

CFE_SB_FULL_SUB_PKT_EID
cfe_sb_eventids.h, 1591

CFE_SB_GET_BUF_ERR_EID
cfe_sb_eventids.h, 1591

CFE_SB_GetPipeByName
cFE Pipe Management APIs, 350

CFE_SB_GETPIPEIDBYNAME_EID
cfe_sb_eventids.h, 1591

CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID
cfe_sb_eventids.h, 1591

CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID

cfe_sb_eventids.h, 1592
CFE_SB_GetPipeName
 cFE Pipe Management APIs, 350
CFE_SB_GETPIPENAME_EID
 cfe_sb_eventids.h, 1592
CFE_SB_GETPIPENAME_ID_ERR_EID
 cfe_sb_eventids.h, 1592
CFE_SB_GETPIPENAME_NULL_PTR_EID
 cfe_sb_eventids.h, 1592
CFE_SB_GetPipeOpts
 cFE Pipe Management APIs, 351
CFE_SB_GETPIPEOPTS_EID
 cfe_sb_eventids.h, 1593
CFE_SB_GETPIPEOPTS_ID_ERR_EID
 cfe_sb_eventids.h, 1593
CFE_SB_GETPIPEOPTS_PTR_ERR_EID
 cfe_sb_eventids.h, 1593
CFE_SB_GetUserData
 cFE Message Characteristics APIs, 364
CFE_SB_GetUserDataLength
 cFE Message Characteristics APIs, 364
CFE_SB_GlobalCmdTopicIdToMsgId
 cFE Message ID APIs, 369
CFE_SB_GlobalTlmTopicIdToMsgId
 cFE Message ID APIs, 370
CFE_SB_HASHCOLLISION_EID
 cfe_sb_eventids.h, 1593
CFE_SB_HK_TLM_MID
 default_cfe_sb_msgids.h, 1579
CFE_SB_HousekeepingTlm, 738
 Payload, 738
 TelemetryHeader, 738
CFE_SB_HousekeepingTlm_Payload, 738
 CommandCounter, 739
 CommandErrorCounter, 739
 CreatePipeErrorCounter, 739
 DuplicateSubscriptionsCounter, 740
 GetPipeByNameErrorCounter, 740
 InternalErrorCounter, 740
 MemInUse, 740
 MemPoolHandle, 740
 MsgLimitErrorCounter, 740
 MsgReceiveErrorCounter, 740
 MsgSendErrorCounter, 741
 NoSubscribersCounter, 741
 PipeOptsErrorCounter, 741
 PipeOverflowErrorCounter, 741
 Spare2Align, 741
 SubscribeErrorCounter, 741
 UnmarkedMem, 741
CFE_SB_HousekeepingTlm_Payload_t
 default_cfe_sb_msgdefs.h, 1577
CFE_SB_HousekeepingTlm_t
 default_cfe_sb_msgstruct.h, 1581
CFE_SB_INIT_EID
 cfe_sb_eventids.h, 1594
CFE_SB_INTERNAL_ERR
 cFE Return Code Defines, 246
CFE_SB_INVALID_MSG_ID
 cfe_sb_api_typedefs.h, 1400
CFE_SB_INVALID_PIPE
 cfe_sb_api_typedefs.h, 1400
CFE_SB_IsValidMsgId
 cFE Message ID APIs, 370
CFE_SB_LEN_ERR_EID
 cfe_sb_eventids.h, 1594
CFE_SB_LocalCmdTopicIdToMsgId
 cFE Message ID APIs, 371
CFE_SB_LocalTlmTopicIdToMsgId
 cFE Message ID APIs, 371
CFE_SB_MAX_DESTS_MET
 cFE Return Code Defines, 246
CFE_SB_MAX_DESTS_MET_EID
 cfe_sb_eventids.h, 1594
CFE_SB_MAX_MSGS_MET
 cFE Return Code Defines, 246
CFE_SB_MAX_MSGS_MET_EID
 cfe_sb_eventids.h, 1594
CFE_SB_MAX_PIPES_MET
 cFE Return Code Defines, 246
CFE_SB_MAX_PIPES_MET_EID
 cfe_sb_eventids.h, 1595
CFE_SB_MessageStringGet
 cFE Message Characteristics APIs, 365
CFE_SB_MessageStringSet
 cFE Message Characteristics APIs, 366
CFE_SB_Msg, 742
 LongDouble, 742
 LongInt, 742
 Msg, 742
CFE_SB_MSG_TOO_BIG
 cFE Return Code Defines, 246
CFE_SB_MSG_TOO_BIG_EID
 cfe_sb_eventids.h, 1595
CFE_SB_MsgId_Atom_t
 default_cfe_sb_extern_typedefs.h, 1555
CFE_SB_MSGID_C
 cfe_sb_api_typedefs.h, 1401
CFE_SB_MsgId_Equal
 cFE Message ID APIs, 371
CFE_SB_MSGID_LIM_ERR_EID
 cfe_sb_eventids.h, 1595
CFE_SB_MSGID_RESERVED
 cfe_sb_api_typedefs.h, 1401
CFE_SB_MsgId_t, 743
 Value, 743
CFE_SB_MSGID_UNWRAP_VALUE
 cfe_sb_api_typedefs.h, 1401

CFE_SB_MSGID_WRAP_VALUE
 cfe_sb_api_typedefs.h, 1401

CFE_SB_MsgIdToValue
 cFE Message ID APIs, 372

CFE_SB_MsgMapFileEntry, 743
 Index, 743
 MsgId, 744

CFE_SB_MsgMapFileEntry_t
 default_cfe_sb_msgdefs.h, 1577

CFE_SB_NO_MESSAGE
 cFE Return Code Defines, 246

CFE_SB_NOOP_CC
 default_cfe_sb_fcncodes.h, 1560

CFE_SB_NoopCmd, 744
 CommandHeader, 744

CFE_SB_NoopCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_NOT_IMPLEMENTED
 cFE Return Code Defines, 247

CFE_SB_ONESUB_TLM_MID
 default_cfe_sb_msgids.h, 1579

CFE_SB_PART_SUB_PKT_EID
 cfe_sb_eventids.h, 1595

CFE_SB_PEND_FOREVER
 cfe_sb_api_typedefs.h, 1401

CFE_SB_PIPE_ADDED_EID
 cfe_sb_eventids.h, 1596

CFE_SB_PIPE_CR_ERR
 cFE Return Code Defines, 247

CFE_SB_PIPE_DELETED_EID
 cfe_sb_eventids.h, 1596

CFE_SB_PIPE_RD_ERR
 cFE Return Code Defines, 247

CFE_SB_PipeDepthStats, 744
 CurrentQueueDepth, 745
 MaxQueueDepth, 745
 PeakQueueDepth, 745
 Pipeld, 745
 Spare, 745

CFE_SB_PipeDepthStats_t
 default_cfe_sb_msgdefs.h, 1577

CFE_SB_PIPEID_BASE
 cFE Resource ID base values, 416

CFE_SB_PIPEID_C
 cfe_sb_api_typedefs.h, 1402

CFE_SB_Pipeld_t
 default_cfe_sb_extern_typedefs.h, 1555

CFE_SB_Pipeld_ToIndex
 cFE Pipe Management APIs, 351

CFE_SB_PipeInfoEntry, 745
 Appld, 746
 AppName, 746
 CurrentQueueDepth, 746
 MaxQueueDepth, 746

 Opts, 746
 PeakQueueDepth, 746
 Pipeld, 747
 PipeName, 747
 SendErrors, 747
 Spare, 747

CFE_SB_PipeInfoEntry_t
 default_cfe_sb_msgdefs.h, 1577

CFE_SB PIPEOPTS_IGNOREMINE
 cFE SB Pipe options, 374

CFE_SB_POLL
 cfe_sb_api_typedefs.h, 1402

CFE_SB_Q_FULL_ERR_EID
 cfe_sb_eventids.h, 1596

CFE_SB_Q_RD_ERR_EID
 cfe_sb_eventids.h, 1596

CFE_SB_Q_WR_ERR_EID
 cfe_sb_eventids.h, 1597

CFE_SB_Qos_t, 747
 Priority, 747
 Reliability, 747

CFE_SB_QosPriority
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosPriority_Enum_t
 default_cfe_sb_extern_typedefs.h, 1555

CFE_SB_QosPriority_HIGH
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosPriority_LOW
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosReliability
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosReliability_Enum_t
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosReliability_HIGH
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_QosReliability_LOW
 default_cfe_sb_extern_typedefs.h, 1556

CFE_SB_RCV_BAD_ARG_EID
 cfe_sb_eventids.h, 1597

CFE_SB_RCV_MESSAGE_INTEGRITY_FAIL_EID
 cfe_sb_eventids.h, 1597

CFE_SB_ReceiveBuffer
 cFE Send/Receive Message APIs, 358

CFE_SB_ReleaseMessageBuffer
 cFE Zero Copy APIs, 361

CFE_SB_RESET_COUNTERS_CC
 default_cfe_sb_fcncodes.h, 1560

CFE_SB_ResetCountersCmd, 748
 CommandHeader, 748

CFE_SB_ResetCountersCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_RouteCmd_Payload, 748
 MsgId, 748
 Pipe, 749

Spare, [749](#)
CFE_SB_RouteCmd_Payload_t
 default_cfe_sb_msgdefs.h, [1577](#)
CFE_SB_Routeld_Atom_t
 default_cfe_sb_extern_typedefs.h, [1556](#)
CFE_SB_RoutingFileEntry, [749](#)
 AppName, [749](#)
 MsgCnt, [749](#)
 MsgId, [750](#)
 PipeId, [750](#)
 PipeName, [750](#)
 State, [750](#)
CFE_SB_RoutingFileEntry_t
 default_cfe_sb_msgdefs.h, [1578](#)
CFE_SB_SEND_BAD_ARG_EID
 cfe_sb_eventids.h, [1597](#)
CFE_SB_SEND_HK_MID
 default_cfe_sb_msgids.h, [1579](#)
CFE_SB_SEND_INV_MSGID_EID
 cfe_sb_eventids.h, [1598](#)
CFE_SB_SEND_MESSAGE_INTEGRITY_FAIL_EID
 cfe_sb_eventids.h, [1598](#)
CFE_SB_SEND_NO_SUBS_EID
 cfe_sb_eventids.h, [1598](#)
CFE_SB_SEND_PREV_SUBS_CC
 default_cfe_sb_fcncodes.h, [1561](#)
CFE_SB_SEND_SB_STATS_CC
 default_cfe_sb_fcncodes.h, [1562](#)
CFE_SB_SendHkCmd, [750](#)
 CommandHeader, [750](#)
CFE_SB_SendHkCmd_t
 default_cfe_sb_msgstruct.h, [1581](#)
CFE_SB_SendPrevSubsCmd, [750](#)
 CommandHeader, [751](#)
CFE_SB_SendPrevSubsCmd_t
 default_cfe_sb_msgstruct.h, [1581](#)
CFE_SB_SendSbStatsCmd, [751](#)
 CommandHeader, [751](#)
CFE_SB_SendSbStatsCmd_t
 default_cfe_sb_msgstruct.h, [1581](#)
CFE_SB_SetPipeOpts
 cFE Pipe Management APIs, [352](#)
CFE_SB_SETPPIPEOPTS_EID
 cfe_sb_eventids.h, [1598](#)
CFE_SB_SETPPIPEOPTS_ID_ERR_EID
 cfe_sb_eventids.h, [1599](#)
CFE_SB_SETPPIPEOPTS_OWNER_ERR_EID
 cfe_sb_eventids.h, [1599](#)
CFE_SB_SetUserDataLength
 cFE Message Characteristics APIs, [367](#)
CFE_SB_SingleSubscriptionTlm, [751](#)
 Payload, [752](#)
 TelemetryHeader, [752](#)
CFE_SB_SingleSubscriptionTlm_Payload, [752](#)
MsgId, [752](#)
Pipe, [752](#)
Qos, [753](#)
SubType, [753](#)
CFE_SB_SingleSubscriptionTlm_Payload_t
 default_cfe_sb_msgdefs.h, [1578](#)
CFE_SB_SingleSubscriptionTlm_t
 default_cfe_sb_msgstruct.h, [1581](#)
CFE_SB SND RTG EID
 cfe_sb_eventids.h, [1599](#)
CFE_SB SND RTG ERR1 EID
 cfe_sb_eventids.h, [1599](#)
CFE_SB SND STATS EID
 cfe_sb_eventids.h, [1600](#)
CFE_SB STATS TLM MID
 default_cfe_sb_msgids.h, [1579](#)
CFE_SB_StatsTlm, [753](#)
 Payload, [753](#)
 TelemetryHeader, [753](#)
CFE_SB_StatsTlm_Payload, [753](#)
 MaxMemAllowed, [754](#)
 MaxMsgIdsAllowed, [754](#)
 MaxPipeDepthAllowed, [755](#)
 MaxPipesAllowed, [755](#)
 MaxSubscriptionsAllowed, [755](#)
 MemInUse, [755](#)
 MsgIdsInUse, [755](#)
 PeakMemInUse, [755](#)
 PeakMsgIdsInUse, [755](#)
 PeakPipesInUse, [756](#)
 PeakSBBuffersInUse, [756](#)
 PeakSubscriptionsInUse, [756](#)
 PipeDepthStats, [756](#)
 PipesInUse, [756](#)
 SBBuffersInUse, [756](#)
 SubscriptionsInUse, [756](#)
CFE_SB_StatsTlm_Payload_t
 default_cfe_sb_msgdefs.h, [1578](#)
CFE_SB_StatsTlm_t
 default_cfe_sb_msgstruct.h, [1581](#)
CFE_SB_SUB_ARG_ERR_EID
 cfe_sb_eventids.h, [1600](#)
CFE_SB_SUB_ENTRIES_PER_PKT
 default_cfe_sb_extern_typedefs.h, [1555](#)
CFE_SB_SUB_INV_CALLER_EID
 cfe_sb_eventids.h, [1600](#)
CFE_SB_SUB_INV_PIPE_EID
 cfe_sb_eventids.h, [1600](#)
CFE_SB_SUB_RPT_CTRL_MID
 default_cfe_sb_msgids.h, [1579](#)
CFE_SB_SubEntries, [757](#)
 MsgId, [757](#)
 Pipe, [757](#)
 Qos, [757](#)

CFE_SB_SubEntries_t
 default_cfe_sb_msgdefs.h, 1578

CFE_SB_Subscribe
 cFE Message Subscription Control APIs, 353

CFE_SB_SubscribeEx
 cFE Message Subscription Control APIs, 354

CFE_SB_SubscribeLocal
 cFE Message Subscription Control APIs, 355

CFE_SB_SUBSCRIPTION
 cfe_sb_api_typedefs.h, 1402

CFE_SB_SUBSCRIPTION_RCVD_EID
 cfe_sb_eventids.h, 1601

CFE_SB_SUBSCRIPTION_REMOVED_EID
 cfe_sb_eventids.h, 1601

CFE_SB_SUBSCRIPTION_RPT_EID
 cfe_sb_eventids.h, 1601

CFE_SB_TIME_OUT
 cFE Return Code Defines, 247

CFE_SB_TimeStampMsg
 cFE Message Characteristics APIs, 367

CFE_SB_TlmTopicIdToMsgId
 cFE Message ID APIs, 372

CFE_SB_TransmitBuffer
 cFE Zero Copy APIs, 362

CFE_SB_TransmitMsg
 cFE Send/Receive Message APIs, 359

CFE_SB_UNSUB_ARG_ERR_EID
 cfe_sb_eventids.h, 1601

CFE_SB_UNSUB_INV_CALLER_EID
 cfe_sb_eventids.h, 1602

CFE_SB_UNSUB_INV_PIPE_EID
 cfe_sb_eventids.h, 1602

CFE_SB_UNSUB_NO_SUBS_EID
 cfe_sb_eventids.h, 1602

CFE_SB_Unsubscribe
 cFE Message Subscription Control APIs, 355

CFE_SB_UnsubscribeLocal
 cFE Message Subscription Control APIs, 356

CFE_SB_UNSUBSCRIPTION
 cfe_sb_api_typedefs.h, 1402

CFE_SB_ValueToMsgId
 cFE Message ID APIs, 373

CFE_SB_WRITE_MAP_INFO_CC
 default_cfe_sb_fcncodes.h, 1563

CFE_SB_WRITE_PIPE_INFO_CC
 default_cfe_sb_fcncodes.h, 1564

CFE_SB_WRITE_ROUTING_INFO_CC
 default_cfe_sb_fcncodes.h, 1565

CFE_SB_WriteFileInfoCmd_Payload, 758
 Filename, 758

CFE_SB_WriteFileInfoCmd_Payload_t
 default_cfe_sb_msgdefs.h, 1578

CFE_SB_WriteMapInfoCmd, 758
 CommandHeader, 758

Payload, 758

CFE_SB_WriteMapInfoCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_WritePipeInfoCmd, 759
 CommandHeader, 759

Payload, 759

CFE_SB_WritePipeInfoCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_WriteRoutingInfoCmd, 759
 CommandHeader, 760

Payload, 760

CFE_SB_WriteRoutingInfoCmd_t
 default_cfe_sb_msgstruct.h, 1581

CFE_SB_WRONG_MSG_TYPE
 cFE Return Code Defines, 247

CFE_SERVICE_BITMASK
 cfe_error.h, 1366

CFE_SET
 cfe_sb.h, 1399

CFE_SEVERITY_BITMASK
 cfe_error.h, 1366

CFE_SEVERITY_ERROR
 cfe_error.h, 1366

CFE_SEVERITY_INFO
 cfe_error.h, 1366

CFE_SEVERITY_SUCCESS
 cfe_error.h, 1366

CFE_SOFTWARE_BUS_SERVICE
 cfe_error.h, 1366

CFE_SRC_VERSION
 cfe_version.h, 1412

CFE_STATUS_BAD_COMMAND_CODE
 cFE Return Code Defines, 247

CFE_STATUS_C
 cfe_error.h, 1366

CFE_STATUS_EXTERNAL_RESOURCE_FAIL
 cFE Return Code Defines, 247

CFE_STATUS_INCORRECT_STATE
 cFE Return Code Defines, 248

CFE_STATUS_NO_COUNTER_INCREMENT
 cFE Return Code Defines, 248

CFE_STATUS_NOT_IMPLEMENTED
 cFE Return Code Defines, 248

CFE_STATUS_RANGE_ERROR
 cFE Return Code Defines, 248

CFE_STATUS_REQUEST_ALREADY_PENDING
 cFE Return Code Defines, 248

CFE_STATUS_STRING_LENGTH
 cfe_error.h, 1366

CFE_Status_t
 cfe_error.h, 1367

CFE_STATUS_UNKNOWN_MSG_ID
 cFE Return Code Defines, 248

CFE_STATUS_VALIDATION_FAILURE

cFE Return Code Defines, [248](#)
CFE_STATUS_WRONG_MSG_LENGTH
 cFE Return Code Defines, [249](#)
CFE_StatusString_t
 `cfe_error.h`, [1367](#)
CFE_STR
 `cfe_version.h`, [1412](#)
CFE_STR_HELPER
 `cfe_version.h`, [1413](#)
CFE_SUCCESS
 cFE Return Code Defines, [249](#)
CFE_TABLE_SERVICE
 `cfe_error.h`, [1366](#)
CFE_TBL_ABORT_LOAD_CC
 `default_cfe_tbl_fcncodes.h`, [1604](#)
CFE_TBL_abortLoadCmd, [760](#)
 CommandHeader, [760](#)
 Payload, [760](#)
CFE_TBL_abortLoadCmd_Payload, [760](#)
 TableName, [761](#)
CFE_TBL_abortLoadCmd_Payload_t
 `default_cfe_tbl_msgdefs.h`, [1622](#)
CFE_TBL_abortLoadCmd_t
 `default_cfe_tbl_msgstruct.h`, [1625](#)
CFE_TBL_ACTIVATE_CC
 `default_cfe_tbl_fcncodes.h`, [1605](#)
CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID
 `cfe_tbl_eventids.h`, [1630](#)
CFE_TBL_ACTIVATE_ERR_EID
 `cfe_tbl_eventids.h`, [1630](#)
CFE_TBL_ActivateCmd, [761](#)
 CommandHeader, [761](#)
 Payload, [761](#)
CFE_TBL_ActivateCmd_Payload, [762](#)
 TableName, [762](#)
CFE_TBL_ActivateCmd_Payload_t
 `default_cfe_tbl_msgdefs.h`, [1622](#)
CFE_TBL_ActivateCmd_t
 `default_cfe_tbl_msgstruct.h`, [1625](#)
cfe_tbl_api_typedefs.h
 CFE_TBL_BAD_TABLE_HANDLE, [1405](#)
 CFE_TBL_CallbackFuncPtr_t, [1405](#)
 CFE_TBL_Handle_t, [1405](#)
 CFE_TBL_Info_t, [1405](#)
 CFE_TBL_MAX_FULL_NAME_LEN, [1405](#)
 CFE_TBL_SRC_ADDRESS, [1406](#)
 CFE_TBL_SRC_FILE, [1406](#)
 CFE_TBL_SrcEnum, [1405](#)
 CFE_TBL_SrcEnum_t, [1405](#)
CFE_TBL_ASSUMED_VALID_INF_EID
 `cfe_tbl_eventids.h`, [1630](#)
CFE_TBL_BAD_ARGUMENT
 cFE Return Code Defines, [249](#)
CFE_TBL_BAD_TABLE_HANDLE

cfe_tbl_api_typedefs.h, [1405](#)
CFE_TBL_BufferSelect
 `default_cfe_tbl_extern_typedefs.h`, [1603](#)
CFE_TBL_BufferSelect_ACTIVE
 `default_cfe_tbl_extern_typedefs.h`, [1604](#)
CFE_TBL_BufferSelect_Enum_t
 `default_cfe_tbl_extern_typedefs.h`, [1603](#)
CFE_TBL_BufferSelect_INACTIVE
 `default_cfe_tbl_extern_typedefs.h`, [1604](#)
CFE_TBL_CallbackFuncPtr_t
 `cfe_tbl_api_typedefs.h`, [1405](#)
CFE_TBL_CC1_ERR_EID
 `cfe_tbl_eventids.h`, [1631](#)
CFE_TBL_CDS_DELETE_ERR_EID
 `cfe_tbl_eventids.h`, [1631](#)
CFE_TBL_CDS_DELETED_INFO_EID
 `cfe_tbl_eventids.h`, [1631](#)
CFE_TBL_CDS_NOT_FOUND_ERR_EID
 `cfe_tbl_eventids.h`, [1631](#)
CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID
 `cfe_tbl_eventids.h`, [1632](#)
CFE_TBL_CMD_MID
 `default_cfe_tbl_msgids.h`, [1623](#)
CFE_TBL_CREATING_DUMP_FILE_ERR_EID
 `cfe_tbl_eventids.h`, [1632](#)
CFE_TBL_DelCDSCmd_Payload, [762](#)
 TableName, [762](#)
CFE_TBL_DelCDSCmd_Payload_t
 `default_cfe_tbl_msgdefs.h`, [1622](#)
CFE_TBL_DELETE_CDS_CC
 `default_cfe_tbl_fcncodes.h`, [1606](#)
CFE_TBL_DeleteCDSCmd, [763](#)
 CommandHeader, [763](#)
 Payload, [763](#)
CFE_TBL_DeleteCDSCmd_t
 `default_cfe_tbl_msgstruct.h`, [1625](#)
CFE_TBL_DUMP_CC
 `default_cfe_tbl_fcncodes.h`, [1607](#)
CFE_TBL_DUMP_PENDING_ERR_EID
 `cfe_tbl_eventids.h`, [1632](#)
CFE_TBL_DUMP_REGISTRY_CC
 `default_cfe_tbl_fcncodes.h`, [1607](#)
CFE_TBL_DumpCmd, [763](#)
 CommandHeader, [763](#)
 Payload, [764](#)
CFE_TBL_DumpCmd_Payload, [764](#)
 ActiveTableFlag, [764](#)
 DumpFilename, [764](#)
 TableName, [764](#)
CFE_TBL_DumpCmd_Payload_t
 `default_cfe_tbl_msgdefs.h`, [1622](#)
CFE_TBL_DumpCmd_t
 `default_cfe_tbl_msgstruct.h`, [1625](#)
CFE_TBL_DUMPCTRLID_BASE

cFE Resource ID base values, [416](#)
CFE_TBL_DumpRegistryCmd, [765](#)
 CommandHeader, [765](#)
 Payload, [765](#)
CFE_TBL_DumpRegistryCmd_Payload, [765](#)
 DumpFilename, [766](#)
CFE_TBL_DumpRegistryCmd_Payload_t
 default_cfe_tbl_msgdefs.h, [1622](#)
CFE_TBL_DumpRegistryCmd_t
 default_cfe_tbl_msgstruct.h, [1625](#)
CFE_TBL_DumpToBuffer
 cFE Manage Table Content APIs, [380](#)
CFE_TBL_ERR_ACCESS
 cFE Return Code Defines, [249](#)
CFE_TBL_ERR_BAD_CONTENT_ID
 cFE Return Code Defines, [249](#)
CFE_TBL_ERR_BAD_PROCESSOR_ID
 cFE Return Code Defines, [249](#)
CFE_TBL_ERR_BAD_SPACECRAFT_ID
 cFE Return Code Defines, [249](#)
CFE_TBL_ERR_BAD_SUBTYPE_ID
 cFE Return Code Defines, [249](#)
CFE_TBL_ERR_DUMP_ONLY
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_DUPLICATE_NOT_OWNED
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_FILE_FOR_WRONG_TABLE
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_FILE_SIZE_INCONSISTENT
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_FILE_TOO_LARGE
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_FILENAME_TOO_LONG
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_HANDLES_FULL
 cFE Return Code Defines, [250](#)
CFE_TBL_ERR_ILLEGAL_SRC_TYPE
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_INVALID_HANDLE
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_INVALID_NAME
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_INVALID_OPTIONS
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_INVALID_SIZE
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_LOAD_IN_PROGRESS
 cFE Return Code Defines, [251](#)
CFE_TBL_ERR_LOAD_INCOMPLETE
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_NEVER_LOADED
 cFE Return Code Defines, [252](#)

CFE_TBL_ERR_NO_ACCESS
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_NO_BUFFER_AVAIL
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_NO_STD_HEADER
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_NO_TBL_HEADER
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_PARTIAL_LOAD
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_REGISTRY_FULL
 cFE Return Code Defines, [252](#)
CFE_TBL_ERR_SHORT_FILE
 cFE Return Code Defines, [253](#)
CFE_TBL_ERR_UNREGISTERED
 cFE Return Code Defines, [253](#)
cfe_tbl_eventids.h
 CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID,
 [1630](#)
 CFE_TBL_ACTIVATE_ERR_EID, [1630](#)
 CFE_TBL_ASSUMED_VALID_INF_EID, [1630](#)
 CFE_TBL_CC1_ERR_EID, [1631](#)
 CFE_TBL_CDS_DELETE_ERR_EID, [1631](#)
 CFE_TBL_CDS_DELETED_INFO_EID, [1631](#)
 CFE_TBL_CDS_NOT_FOUND_ERR_EID, [1631](#)
 CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID, [1632](#)
 CFE_TBL_CREATING_DUMP_FILE_ERR_EID,
 [1632](#)
 CFE_TBL_DUMP_PENDING_ERR_EID, [1632](#)
 CFE_TBL_FAIL_HK_SEND_ERR_EID, [1632](#)
 CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID, [1633](#)
 CFE_TBL_FILE_ACCESS_ERR_EID, [1633](#)
 CFE_TBL_FILE_INCOMPLETE_ERR_EID, [1633](#)
 CFE_TBL_FILE_LOADED_INF_EID, [1633](#)
 CFE_TBL_FILE_STD_HDR_ERR_EID, [1634](#)
 CFE_TBL_FILE_SUBTYPE_ERR_EID, [1634](#)
 CFE_TBL_FILE_TBL_HDR_ERR_EID, [1634](#)
 CFE_TBL_FILE_TOO_BIG_ERR_EID, [1634](#)
 CFE_TBL_FILE_TYPE_ERR_EID, [1635](#)
 CFE_TBL_HANDLE_ACCESS_ERR_EID, [1635](#)
 CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID,
 [1635](#)
 CFE_TBL_IN_REGISTRY_ERR_EID, [1635](#)
 CFE_TBL_INIT_INF_EID, [1636](#)
 CFE_TBL_LEN_ERR_EID, [1636](#)
 CFE_TBL_LOAD_ABORT_ERR_EID, [1636](#)
 CFE_TBL_LOAD_ABORT_INF_EID, [1636](#)
 CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID, [1637](#)
 CFE_TBL_LOAD_FILENAME_LONG_ERR_EID,
 [1637](#)
 CFE_TBL_LOAD_IN_PROGRESS_ERR_EID, [1637](#)
 CFE_TBL_LOAD_PEND_REQ_INF_EID, [1637](#)
 CFE_TBL_LOAD_SUCCESS_INF_EID, [1638](#)

CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID, [1638](#)
 CFE_TBL_LOAD_TYPE_ERR_EID, [1638](#)
 CFE_TBL_LOAD_VAL_ERR_EID, [1638](#)
 CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID, [1639](#)
 CFE_TBL_LOADING_PENDING_ERR_EID, [1639](#)
 CFE_TBL_MID_ERR_EID, [1639](#)
 CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID, [1639](#)
 CFE_TBL_NO_SUCH_TABLE_ERR_EID, [1640](#)
 CFE_TBL_NO_WORK_BUFFERS_ERR_EID, [1640](#)
 CFE_TBL_NOOP_INF_EID, [1640](#)
 CFE_TBL_NOT_CRITICAL_TBL_ERR_EID, [1640](#)
 CFE_TBL_NOT_IN_CRIT_REG_ERR_EID, [1641](#)
 CFE_TBL_OVERWRITE_DUMP_INF_EID, [1641](#)
 CFE_TBL_OVERWRITE_REG_DUMP_INF_EID, [1641](#)
 CFE_TBL_PARTIAL_LOAD_ERR_EID, [1641](#)
 CFE_TBL_PROCESSOR_ID_ERR_EID, [1642](#)
 CFE_TBL_REGISTER_ERR_EID, [1642](#)
 CFE_TBL_RESET_INF_EID, [1642](#)
 CFE_TBL_SHARE_ERR_EID, [1642](#)
 CFE_TBL_SPACECRAFT_ID_ERR_EID, [1643](#)
 CFE_TBL_TLM_REG_CMD_INF_EID, [1643](#)
 CFE_TBL_TOO_MANY_DUMPS_ERR_EID, [1643](#)
 CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID, [1643](#)
 CFE_TBL_UNREGISTER_ERR_EID, [1644](#)
 CFE_TBL_UNVALIDATED_ERR_EID, [1644](#)
 CFE_TBL_UPDATE_ERR_EID, [1644](#)
 CFE_TBL_UPDATE_SUCCESS_INF_EID, [1644](#)
 CFE_TBL_VAL_REQ_MADE_INF_EID, [1645](#)
 CFE_TBL_VALIDATION_ERR_EID, [1645](#)
 CFE_TBL_VALIDATION_INF_EID, [1645](#)
 CFE_TBL_WRITE_CFE_HDR_ERR_EID, [1645](#)
 CFE_TBL_WRITE_DUMP_INF_EID, [1646](#)
 CFE_TBL_WRITE_REG_DUMP_INF_EID, [1646](#)
 CFE_TBL_WRITE_TBL_HDR_ERR_EID, [1646](#)
 CFE_TBL_WRITE_TBL_IMG_ERR_EID, [1646](#)
 CFE_TBL_WRITE_TBL_REG_ERR_EID, [1647](#)
 CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID, [1647](#)
 CFE_TBL_FAIL_HK_SEND_ERR_EID
 cfe_tbl_eventids.h, [1632](#)
 CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID
 cfe_tbl_eventids.h, [1633](#)
 CFE_TBL_FILE_ACCESS_ERR_EID
 cfe_tbl_eventids.h, [1633](#)
 CFE_TBL_File_Hdr, [766](#)
 NumBytes, [766](#)
 Offset, [766](#)
 Reserved, [766](#)
 TableName, [766](#)
 CFE_TBL_File_Hdr_t
 default_cfe_tbl_extern_typedefs.h, [1603](#)
 CFE_TBL_FILE_INCOMPLETE_ERR_EID
 cfe_tbl_eventids.h, [1633](#)
 CFE_TBL_FILE_LOADED_INF_EID
 cfe_tbl_eventids.h, [1633](#)
 CFE_TBL_FILE_STD_HDR_ERR_EID
 cfe_tbl_eventids.h, [1634](#)
 CFE_TBL_FILE_SUBTYPE_ERR_EID
 cfe_tbl_eventids.h, [1634](#)
 CFE_TBL_FILE_TBL_HDR_ERR_EID
 cfe_tbl_eventids.h, [1634](#)
 CFE_TBL_FILE_TOO_BIG_ERR_EID
 cfe_tbl_eventids.h, [1634](#)
 CFE_TBL_FILE_TYPE_ERR_EID
 cfe_tbl_eventids.h, [1635](#)
 CFE_TBL_FILEDEF
 cfe_tbl_filedef.h, [1406](#)
 CFE_TBL_FileDef, [767](#)
 Description, [767](#)
 ObjectName, [767](#)
 ObjectSize, [768](#)
 TableName, [768](#)
 TgtFilename, [768](#)
 cfe_tbl_filedef.h
 CFE_TBL_FILEDEF, [1406](#)
 CFE_TBL_FileDef_t, [1407](#)
 CFE_TBL_FileDef_t
 cfe_tbl_filedef.h, [1407](#)
 CFE_TBL_GetAddress
 cFE Access Table Content APIs, [386](#)
 CFE_TBL_GetAddresses
 cFE Access Table Content APIs, [387](#)
 CFE_TBL_GetInfo
 cFE Get Table Information APIs, [391](#)
 CFE_TBL_GetStatus
 cFE Get Table Information APIs, [392](#)
 CFE_TBL_HANDLE_ACCESS_ERR_EID
 cfe_tbl_eventids.h, [1635](#)
 CFE_TBL_Handle_t
 cfe_tbl_api_typedefs.h, [1405](#)
 CFE_TBL_HK_TLM_MID
 default_cfe_tbl_msgids.h, [1623](#)
 CFE_TBL_HousekeepingTlm, [768](#)
 Payload, [768](#)
 TelemetryHeader, [768](#)
 CFE_TBL_HousekeepingTlm_Payload, [769](#)
 ActiveBuffer, [770](#)
 ByteAlignPad1, [770](#)
 CommandCounter, [770](#)
 CommandErrorCounter, [770](#)
 FailedValCounter, [770](#)
 LastFileDumped, [770](#)
 LastFileLoaded, [771](#)
 LastTableLoaded, [771](#)

LastUpdatedTable, 771
LastUpdateTime, 771
LastValCrc, 771
LastValStatus, 771
LastValTableName, 771
MemPoolHandle, 772
NumFreeSharedBufs, 772
NumLoadPending, 772
NumTables, 772
NumValRequests, 772
SuccessValCounter, 772
ValidationCounter, 772
CFE_TBL_HousekeepingTlm_Payload_t
 default_cfe_tbl_msgdefs.h, 1622
CFE_TBL_HousekeepingTlm_t
 default_cfe_tbl_msgstruct.h, 1625
CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID
 cfe_tbl_eventids.h, 1635
CFE_TBL_IN_REGISTRY_ERR_EID
 cfe_tbl_eventids.h, 1635
CFE_TBL_Info, 773
 Crc, 773
 Critical, 773
 DoubleBuffered, 774
 DumpOnly, 774
 FileTime, 774
 LastFileLoaded, 774
 NumUsers, 774
 Size, 774
 TableLoadedOnce, 774
 TimeOfLastUpdate, 774
 UserDefAddr, 774
CFE_TBL_INFO_DUMP_PENDING
 cFE Return Code Defines, 253
CFE_TBL_INFO_NO_UPDATE_PENDING
 cFE Return Code Defines, 253
CFE_TBL_INFO_NO_VALIDATION_PENDING
 cFE Return Code Defines, 253
CFE_TBL_INFO_RECOVERED_TBL
 cFE Return Code Defines, 253
CFE_TBL_Info_t
 cfe_tbl_api_typedefs.h, 1405
CFE_TBL_INFO_TABLE_LOCKED
 cFE Return Code Defines, 253
CFE_TBL_INFO_UPDATE_PENDING
 cFE Return Code Defines, 253
CFE_TBL_INFO_UPDATED
 cFE Return Code Defines, 254
CFE_TBL_INFO_VALIDATION_PENDING
 cFE Return Code Defines, 254
CFE_TBL_INIT_INF_EID
 cfe_tbl_eventids.h, 1636
CFE_TBL_LEN_ERR_EID
 cfe_tbl_eventids.h, 1636
CFE_TBL_Load
 cFE Manage Table Content APIs, 381
CFE_TBL_LOAD_ABORT_ERR_EID
 cfe_tbl_eventids.h, 1636
CFE_TBL_LOAD_ABORT_INF_EID
 cfe_tbl_eventids.h, 1636
CFE_TBL_LOAD_CC
 default_cfe_tbl_fcncodes.h, 1608
CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID
 cfe_tbl_eventids.h, 1637
CFE_TBL_LOAD_FILENAME_LONG_ERR_EID
 cfe_tbl_eventids.h, 1637
CFE_TBL_LOAD_IN_PROGRESS_ERR_EID
 cfe_tbl_eventids.h, 1637
CFE_TBL_LOAD_PEND_REQ_INF_EID
 cfe_tbl_eventids.h, 1637
CFE_TBL_LOAD_SUCCESS_INF_EID
 cfe_tbl_eventids.h, 1638
CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID
 cfe_tbl_eventids.h, 1638
CFE_TBL_LOAD_TYPE_ERR_EID
 cfe_tbl_eventids.h, 1638
CFE_TBL_LOAD_VAL_ERR_EID
 cfe_tbl_eventids.h, 1638
CFE_TBL_LoadCmd, 775
 CommandHeader, 775
 Payload, 775
CFE_TBL_LoadCmd_Payload, 775
 LoadFilename, 775
CFE_TBL_LoadCmd_Payload_t
 default_cfe_tbl_msgdefs.h, 1622
CFE_TBL_LoadCmd_t
 default_cfe_tbl_msgstruct.h, 1625
CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID
 cfe_tbl_eventids.h, 1639
CFE_TBL_LOADING_PENDING_ERR_EID
 cfe_tbl_eventids.h, 1639
CFE_TBL_Manage
 cFE Manage Table Content APIs, 382
CFE_TBL_MAX_FULL_NAME_LEN
 cfe_tbl_api_typedefs.h, 1405
CFE_TBL_MESSAGE_ERROR
 cFE Return Code Defines, 254
CFE_TBL_MID_ERR_EID
 cfe_tbl_eventids.h, 1639
CFE_TBL_Modified
 cFE Manage Table Content APIs, 383
CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID
 cfe_tbl_eventids.h, 1639
CFE_TBL_NO SUCH_TABLE_ERR_EID
 cfe_tbl_eventids.h, 1640
CFE_TBL_NO_WORK_BUFFERS_ERR_EID
 cfe_tbl_eventids.h, 1640
CFE_TBL_NOOP_CC

default_cfe_tbl_fcncodes.h, 1609
CFE_TBL_NOOP_INF_EID
 cfe_tbl_eventids.h, 1640
CFE_TBL_NoopCmd, 776
 CommandHeader, 776
CFE_TBL_NoopCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_NOT_CRITICAL_TBL_ERR_EID
 cfe_tbl_eventids.h, 1640
CFE_TBL_NOT_IMPLEMENTED
 cFE Return Code Defines, 254
CFE_TBL_NOT_IN_CRIT_REG_ERR_EID
 cfe_tbl_eventids.h, 1641
CFE_TBL_NotifyByMessage
 cFE Get Table Information APIs, 392
CFE_TBL_NotifyCmd, 776
 CommandHeader, 776
 Payload, 776
CFE_TBL_NotifyCmd_Payload, 777
 Parameter, 777
CFE_TBL_NotifyCmd_Payload_t
 default_cfe_tbl_msgdefs.h, 1622
CFE_TBL_NotifyCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_OPT_BUFFER_MSK
 cFE Table Type Defines, 394
CFE_TBL_OPT_CRITICAL
 cFE Table Type Defines, 394
CFE_TBL_OPT_CRITICAL_MSK
 cFE Table Type Defines, 394
CFE_TBL_OPT_DBL_BUFFER
 cFE Table Type Defines, 394
CFE_TBL_OPT_DEFAULT
 cFE Table Type Defines, 395
CFE_TBL_OPT_DUMP_ONLY
 cFE Table Type Defines, 395
CFE_TBL_OPT_LD_DMP_MSK
 cFE Table Type Defines, 395
CFE_TBL_OPT_LOAD_DUMP
 cFE Table Type Defines, 395
CFE_TBL_OPT_NOT_CRITICAL
 cFE Table Type Defines, 395
CFE_TBL_OPT_NOT_USR_DEF
 cFE Table Type Defines, 395
CFE_TBL_OPT_SNGL_BUFFER
 cFE Table Type Defines, 395
CFE_TBL_OPT_USR_DEF_ADDR
 cFE Table Type Defines, 395
CFE_TBL_OPT_USR_DEF_MSK
 cFE Table Type Defines, 395
CFE_TBL_OVERWRITE_DUMP_INF_EID
 cfe_tbl_eventids.h, 1641
CFE_TBL_OVERWRITE_REG_DUMP_INF_EID
 cfe_tbl_eventids.h, 1641
CFE_TBL_PARTIAL_LOAD_ERR_EID
 cfe_tbl_eventids.h, 1641
CFE_TBL_PROCESSOR_ID_ERR_EID
 cfe_tbl_eventids.h, 1642
CFE_TBL_REG_TLM_MID
 default_cfe_tbl_msgids.h, 1624
CFE_TBL_Register
 cFE Registration APIs, 375
CFE_TBL_REGISTER_ERR_EID
 cfe_tbl_eventids.h, 1642
CFE_TBL_ReleaseAddress
 cFE Access Table Content APIs, 388
CFE_TBL_ReleaseAddresses
 cFE Access Table Content APIs, 389
CFE_TBL_RESET_COUNTERS_CC
 default_cfe_tbl_fcncodes.h, 1610
CFE_TBL_RESET_INF_EID
 cfe_tbl_eventids.h, 1642
CFE_TBL_ResetCountersCmd, 777
 CommandHeader, 777
CFE_TBL_ResetCountersCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_SEND_HK_MID
 default_cfe_tbl_msgids.h, 1624
CFE_TBL_SEND_REGISTRY_CC
 default_cfe_tbl_fcncodes.h, 1611
CFE_TBL_SendHkCmd, 778
 CommandHeader, 778
CFE_TBL_SendHkCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_SendRegistryCmd, 778
 CommandHeader, 778
 Payload, 779
CFE_TBL_SendRegistryCmd_Payload, 779
 TableName, 779
CFE_TBL_SendRegistryCmd_Payload_t
 default_cfe_tbl_msgdefs.h, 1623
CFE_TBL_SendRegistryCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_Share
 cFE Registration APIs, 377
CFE_TBL_SHARE_ERR_EID
 cfe_tbl_eventids.h, 1642
CFE_TBL_SPACECRAFT_ID_ERR_EID
 cfe_tbl_eventids.h, 1643
CFE_TBL_SRC_ADDRESS
 cfe_tbl_api_typedefs.h, 1406
CFE_TBL_SRC_FILE
 cfe_tbl_api_typedefs.h, 1406
CFE_TBL_SrcEnum
 cfe_tbl_api_typedefs.h, 1405
CFE_TBL_SrcEnum_t
 cfe_tbl_api_typedefs.h, 1405
CFE_TBL_TableRegistryTlm, 779

Payload, 780
TelemetryHeader, 780
CFE_TBL_TableRegistryTlm_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_TblRegPacket_Payload, 780
 ActiveBufferAddr, 781
 ByteAlign4, 781
 Crc, 781
 Critical, 781
 DoubleBuffered, 781
 DumpOnly, 781
 FileTime, 782
 InactiveBufferAddr, 782
 LastFileLoaded, 782
 LoadPending, 782
 Name, 782
 OwnerAppName, 782
 Size, 782
 TableLoadedOnce, 783
 TimeOfLastUpdate, 783
 ValidationFuncPtr, 783
CFE_TBL_TblRegPacket_Payload_t
 default_cfe_tbl_msgdefs.h, 1623
CFE_TBL_TLM_REG_CMD_INF_EID
 cfe_tbl_eventids.h, 1643
CFE_TBL_TOO_MANY_DUMPSS_ERR_EID
 cfe_tbl_eventids.h, 1643
CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID
 cfe_tbl_eventids.h, 1643
CFE_TBL_Unregister
 cFE Registration APIs, 378
CFE_TBL_UNREGISTER_ERR_EID
 cfe_tbl_eventids.h, 1644
CFE_TBL_UNVALIDATED_ERR_EID
 cfe_tbl_eventids.h, 1644
CFE_TBL_Update
 cFE Manage Table Content APIs, 384
CFE_TBL_UPDATE_ERR_EID
 cfe_tbl_eventids.h, 1644
CFE_TBL_UPDATE_SUCCESS_INF_EID
 cfe_tbl_eventids.h, 1644
CFE_TBL_VAL_REQ_MADE_INF_EID
 cfe_tbl_eventids.h, 1645
CFE_TBL_Validate
 cFE Manage Table Content APIs, 384
CFE_TBL_VALIDATE_CC
 default_cfe_tbl_fncodes.h, 1612
CFE_TBL_ValidateCmd, 783
 CommandHeader, 783
 Payload, 784
CFE_TBL_ValidateCmd_Payload, 784
 ActiveTableFlag, 784
 TableName, 784
CFE_TBL_ValidateCmd_Payload_t
 default_cfe_tbl_msgdefs.h, 1623
CFE_TBL_ValidateCmd_t
 default_cfe_tbl_msgstruct.h, 1626
CFE_TBL_VALIDATION_ERR_EID
 cfe_tbl_eventids.h, 1645
CFE_TBL_VALIDATION_INF_EID
 cfe_tbl_eventids.h, 1645
CFE_TBL_VALRESULTID_BASE
 cFE Resource ID base values, 416
CFE_TBL_WARN_DUPLICATE
 cFE Return Code Defines, 254
CFE_TBL_WARN_NOT_CRITICAL
 cFE Return Code Defines, 254
CFE_TBL_WARN_PARTIAL_LOAD
 cFE Return Code Defines, 254
CFE_TBL_WARN_SHORT_FILE
 cFE Return Code Defines, 255
CFE_TBL_WRITE_CFE_HDR_ERR_EID
 cfe_tbl_eventids.h, 1645
CFE_TBL_WRITE_DUMP_INF_EID
 cfe_tbl_eventids.h, 1646
CFE_TBL_WRITE_REG_DUMP_INF_EID
 cfe_tbl_eventids.h, 1646
CFE_TBL_WRITE_TBL_HDR_ERR_EID
 cfe_tbl_eventids.h, 1646
CFE_TBL_WRITE_TBL_IMG_ERR_EID
 cfe_tbl_eventids.h, 1646
CFE_TBL_WRITE_TBL_REG_ERR_EID
 cfe_tbl_eventids.h, 1647
CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID
 cfe_tbl_eventids.h, 1647
cfe_time.h
 CFE_TIME_Copy, 1409
CFE_TIME_1HZ_CMD_MID
 default_cfe_time_msgids.h, 1681
CFE_TIME_A_GT_B
 cfe_time_api_typedefs.h, 1410
CFE_TIME_A_LT_B
 cfe_time_api_typedefs.h, 1410
CFE_TIME_Add
 cFE Time Arithmetic APIs, 402
CFE_TIME_ADD_ADJUST_CC
 default_cfe_time_fncodes.h, 1653
CFE_TIME_ADD_DELAY_CC
 default_cfe_time_fncodes.h, 1653
CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC
 default_cfe_time_fncodes.h, 1654
CFE_TIME_AddAdjustCmd, 784
 CommandHeader, 785
 Payload, 785
CFE_TIME_AddAdjustCmd_t
 default_cfe_time_msgstruct.h, 1683
CFE_TIME_AddDelayCmd, 785
 CommandHeader, 785

Payload, 785
 CFE_TIME_AddDelayCmd_t
 default_cfe_time_msgstruct.h, 1683
 CFE_TIME_AddOneHzAdjustmentCmd, 786
 CommandHeader, 786
 Payload, 786
 CFE_TIME_AddOneHzAdjustmentCmd_t
 default_cfe_time_msgstruct.h, 1683
 CFE_TIME_AdjustDirection
 default_cfe_time_extern_typedefs.h, 1650
 CFE_TIME_AdjustDirection_ADD
 default_cfe_time_extern_typedefs.h, 1650
 CFE_TIME_AdjustDirection_Enum_t
 default_cfe_time_extern_typedefs.h, 1649
 CFE_TIME_AdjustDirection_SUBTRACT
 default_cfe_time_extern_typedefs.h, 1650
 cfe_time_api_typedefs.h
 CFE_TIME_A_GT_B, 1410
 CFE_TIME_A_LT_B, 1410
 CFE_TIME_Compare, 1410
 CFE_TIME_Compare_t, 1410
 CFE_TIME_EQUAL, 1410
 CFE_TIME_PRINTED_STRING_SIZE, 1409
 CFE_TIME_SynchCallbackPtr_t, 1410
 CFE_TIME_ZERO_VALUE, 1409
 CFE_TIME_BAD_ARGUMENT
 cFE Return Code Defines, 255
 CFE_TIME_CALLBACK_NOT_REGISTERED
 cFE Return Code Defines, 255
 CFE_TIME_CC_ERR_EID
 cfe_time_eventids.h, 1688
 CFE_TIME_ClockState
 default_cfe_time_extern_typedefs.h, 1650
 CFE_TIME_ClockState_Enum_t
 default_cfe_time_extern_typedefs.h, 1649
 CFE_TIME_ClockState_FLYWHEEL
 default_cfe_time_extern_typedefs.h, 1651
 CFE_TIME_ClockState_INVALID
 default_cfe_time_extern_typedefs.h, 1651
 CFE_TIME_ClockState_VALID
 default_cfe_time_extern_typedefs.h, 1651
 CFE_TIME_CMD_MID
 default_cfe_time_msgids.h, 1681
 CFE_TIME_Compare
 cFE Time Arithmetic APIs, 402
 cfe_time_api_typedefs.h, 1410
 CFE_TIME_Compare_t
 cfe_time_api_typedefs.h, 1410
 CFE_TIME_Copy
 cfe_time.h, 1409
 CFE_TIME_DATA_CMD_MID
 default_cfe_time_msgids.h, 1681
 CFE_TIME_DELAY_CFG_EID
 cfe_time_eventids.h, 1688
 CFE_TIME_DELAY_EID
 cfe_time_eventids.h, 1689
 CFE_TIME_DELAY_ERR_EID
 cfe_time_eventids.h, 1689
 CFE_TIME_DELTA_CFG_EID
 cfe_time_eventids.h, 1689
 CFE_TIME_DELTA_EID
 cfe_time_eventids.h, 1689
 CFE_TIME_DELTA_ERR_EID
 cfe_time_eventids.h, 1690
 CFE_TIME_DIAG_EID
 cfe_time_eventids.h, 1690
 CFE_TIME_DIAG_TLM_MID
 default_cfe_time_msgids.h, 1681
 CFE_TIME_DiagnosticTlm, 786
 Payload, 787
 TelemetryHeader, 787
 CFE_TIME_DiagnosticTlm_Payload, 787
 AtToneDelay, 789
 AtToneLatch, 789
 AtToneLeapSeconds, 789
 AtToneMET, 789
 AtToneSTCF, 789
 ClockFlyState, 790
 ClockSetState, 790
 ClockSignal, 790
 ClockSource, 790
 ClockStateAPI, 790
 ClockStateFlags, 790
 CurrentLatch, 790
 CurrentMET, 791
 CurrentTAI, 791
 CurrentUTC, 791
 DataStoreStatus, 791
 DelayDirection, 791
 Forced2Fly, 791
 LocalIntCounter, 791
 LocalTaskCounter, 792
 MaxElapsed, 792
 MaxLocalClock, 792
 MinElapsed, 792
 OneHzAdjust, 792
 OneHzDirection, 792
 OneTimeAdjust, 792
 OneTimeDirection, 793
 ServerFlyState, 793
 TimeSinceTone, 793
 ToneDataCounter, 793
 ToneDataLatch, 793
 ToneIntCounter, 793
 ToneIntErrorCounter, 793
 ToneMatchCounter, 794
 ToneMatchErrorCounter, 794
 ToneOverLimit, 794

ToneSignalCounter, 794
ToneSignalLatch, 794
ToneTaskCounter, 794
ToneUnderLimit, 794
VersionCounter, 795
VirtualMET, 795
CFE_TIME_DiagnosticTim_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_DiagnosticTim_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_EQUAL
 cfe_time_api_typedefs.h, 1410
cfe_time_eventids.h
 CFE_TIME_CC_ERR_EID, 1688
 CFE_TIME_DELAY_CFG_EID, 1688
 CFE_TIME_DELAY_EID, 1689
 CFE_TIME_DELAY_ERR_EID, 1689
 CFE_TIME_DELTA_CFG_EID, 1689
 CFE_TIME_DELTA_EID, 1689
 CFE_TIME_DELTA_ERR_EID, 1690
 CFE_TIME_DIAG_EID, 1690
 CFE_TIME_FLY_OFF_EID, 1690
 CFE_TIME_FLY_ON_EID, 1690
 CFE_TIME_ID_ERR_EID, 1691
 CFE_TIME_INIT_EID, 1691
 CFE_TIME_LEAPS_CFG_EID, 1691
 CFE_TIME_LEAPS_EID, 1691
 CFE_TIME_LEN_ERR_EID, 1692
 CFE_TIME_MET_CFG_EID, 1692
 CFE_TIME_MET_EID, 1692
 CFE_TIME_MET_ERR_EID, 1692
 CFE_TIME_NOOP_EID, 1693
 CFE_TIME_ONEHZ_CFG_EID, 1693
 CFE_TIME_ONEHZ_EID, 1693
 CFE_TIME_RESET_EID, 1693
 CFE_TIME_SIGNAL_CFG_EID, 1694
 CFE_TIME_SIGNAL_EID, 1694
 CFE_TIME_SIGNAL_ERR_EID, 1694
 CFE_TIME_SOURCE_CFG_EID, 1694
 CFE_TIME_SOURCE_EID, 1695
 CFE_TIME_SOURCE_ERR_EID, 1695
 CFE_TIME_STATE_EID, 1695
 CFE_TIME_STATE_ERR_EID, 1695
 CFE_TIME_STCF_CFG_EID, 1696
 CFE_TIME_STCF_EID, 1696
 CFE_TIME_STCF_ERR_EID, 1696
 CFE_TIME_TIME_CFG_EID, 1696
 CFE_TIME_TIME_EID, 1697
 CFE_TIME_TIME_ERR_EID, 1697
CFE_TIME_ExternalGPS
 cFE External Time Source APIs, 407
CFE_TIME_ExternalMET
 cFE External Time Source APIs, 408
CFE_TIME_ExternalTime
 cFE External Time Source APIs, 408
CFE_TIME_ExternalTone
 cFE External Time Source APIs, 409
CFE_TIME_FakeToneCmd, 795
 CommandHeader, 795
CFE_TIME_FakeToneCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_FLAG_ADD1HZ
 cFE Clock State Flag Defines, 417
CFE_TIME_FLAG_ADDADJ
 cFE Clock State Flag Defines, 417
CFE_TIME_FLAG_ADDTCL
 cFE Clock State Flag Defines, 417
CFE_TIME_FLAG_CLKSET
 cFE Clock State Flag Defines, 417
CFE_TIME_FLAG_CMDFLY
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_FLYING
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_GDTONE
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_REFERR
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_SERVER
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_SIGPRI
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_SRCINT
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_SRVFLY
 cFE Clock State Flag Defines, 418
CFE_TIME_FLAG_UNUSED
 cFE Clock State Flag Defines, 418
CFE_TIME_FlagBit
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_ADD1HZ
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_ADDADJ
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_ADDTCL
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_CLKSET
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_CMDFLY
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_Enum_t
 default_cfe_time_extern_typedefs.h, 1649
CFE_TIME_FlagBit_FLYING
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_GDTONE
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_SERVER
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_SIGPRI

default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_SRCINT
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlagBit_SRVFLY
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FLY_OFF_EID
 cfe_time_eventids.h, 1690
CFE_TIME_FLY_ON_EID
 cfe_time_eventids.h, 1690
CFE_TIME_FlywheelState
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlywheelState_Enum_t
 default_cfe_time_extern_typedefs.h, 1649
CFE_TIME_FlywheelState_IS_FLY
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_FlywheelState_NO_FLY
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_GetClockInfo
 cFE Get Time Information APIs, 399
CFE_TIME_GetClockState
 cFE Get Time Information APIs, 399
CFE_TIME_GetLeapSeconds
 cFE Get Time Information APIs, 400
CFE_TIME_GetMET
 cFE Get Current Time APIs, 396
CFE_TIME_GetMETseconds
 cFE Get Current Time APIs, 396
CFE_TIME_GetMETsubsecs
 cFE Get Current Time APIs, 397
CFE_TIME_GetSTCF
 cFE Get Time Information APIs, 400
CFE_TIME_GetTAI
 cFE Get Current Time APIs, 397
CFE_TIME_GetTime
 cFE Get Current Time APIs, 398
CFE_TIME_GetUTC
 cFE Get Current Time APIs, 398
CFE_TIME_HK_TLM_MID
 default_cfe_time_msgids.h, 1681
CFE_TIME_HousekeepingTlm, 795
 Payload, 796
 TelemetryHeader, 796
CFE_TIME_HousekeepingTlm_Payload, 796
 ClockStateAPI, 797
 ClockStateFlags, 797
 CommandCounter, 797
 CommandErrorCounter, 797
 LeapSeconds, 797
 Seconds1HzAdj, 798
 SecondsDelay, 798
 SecondsMET, 798
 SecondsSTCF, 798
 Subsecs1HzAdj, 798
 SubsecsDelay, 798
 SubsecsMET, 798
 SubsecsSTCF, 799
CFE_TIME_HousekeepingTlm_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_HousekeepingTlm_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_ID_ERR_EID
 cfe_time_eventids.h, 1691
CFE_TIME_INIT_EID
 cfe_time_eventids.h, 1691
CFE_TIME_INTERNAL_ONLY
 cFE Return Code Defines, 255
CFE_TIME_LEAPS_CFG_EID
 cfe_time_eventids.h, 1691
CFE_TIME_LEAPS_EID
 cfe_time_eventids.h, 1691
CFE_TIME_LeapsCmd_Payload, 799
 LeapSeconds, 799
CFE_TIME_LeapsCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_LEN_ERR_EID
 cfe_time_eventids.h, 1692
CFE_TIME_Local1HzISR
 cFE Miscellaneous Time APIs, 412
CFE_TIME_MET2SCTime
 cFE Time Conversion APIs, 405
CFE_TIME_MET_CFG_EID
 cfe_time_eventids.h, 1692
CFE_TIME_MET_EID
 cfe_time_eventids.h, 1692
CFE_TIME_MET_ERR_EID
 cfe_time_eventids.h, 1692
CFE_TIME_Micro2SubSecs
 cFE Time Conversion APIs, 405
CFE_TIME_NOOP_CC
 default_cfe_time_fnccodes.h, 1655
CFE_TIME_NOOP_EID
 cfe_time_eventids.h, 1693
CFE_TIME_NoopCmd, 799
 CommandHeader, 800
CFE_TIME_NoopCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_NOT_IMPLEMENTED
 cFE Return Code Defines, 255
CFE_TIME_ONEHZ_CFG_EID
 cfe_time_eventids.h, 1693
CFE_TIME_ONEHZ_CMD_MID
 default_cfe_time_msgids.h, 1681
CFE_TIME_ONEHZ_EID
 cfe_time_eventids.h, 1693
CFE_TIME_OneHzAdjustmentCmd_Payload, 800
 Seconds, 800
 Subseconds, 800
CFE_TIME_OneHzAdjustmentCmd_Payload_t

default_cfe_time_msgdefs.h, 1680
CFE_TIME_OneHzCmd, 800
 CommandHeader, 800
CFE_TIME_OneHzCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_OUT_OF_RANGE
 cFE Return Code Defines, 255
CFE_TIME_Print
 cFE Miscellaneous Time APIs, 412
CFE_TIME_PRINTED_STRING_SIZE
 cfe_time_api_typedefs.h, 1409
CFE_TIME_RegisterSynchCallback
 cFE External Time Source APIs, 409
CFE_TIME_RESET_COUNTERS_CC
 default_cfe_time_fncodes.h, 1656
CFE_TIME_RESET_EID
 cfe_time_eventids.h, 1693
CFE_TIME_ResetCountersCmd, 801
 CommandHeader, 801
CFE_TIME_ResetCountersCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SEND_CMD_MID
 default_cfe_time_msgids.h, 1682
CFE_TIME_SEND_DIAGNOSTIC_CC
 default_cfe_time_fncodes.h, 1657
CFE_TIME_SEND_HK_MID
 default_cfe_time_msgids.h, 1682
CFE_TIME_SendDiagnosticCmd, 801
 CommandHeader, 801
CFE_TIME_SendDiagnosticCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SendHkCmd, 802
 CommandHeader, 802
CFE_TIME_SendHkCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SERVICE
 cfe_error.h, 1367
CFE_TIME_SET_LEAP_SECONDS_CC
 default_cfe_time_fncodes.h, 1658
CFE_TIME_SET_MET_CC
 default_cfe_time_fncodes.h, 1659
CFE_TIME_SET_SIGNAL_CC
 default_cfe_time_fncodes.h, 1660
CFE_TIME_SET_SOURCE_CC
 default_cfe_time_fncodes.h, 1660
CFE_TIME_SET_STATE_CC
 default_cfe_time_fncodes.h, 1661
CFE_TIME_SET_STCF_CC
 default_cfe_time_fncodes.h, 1663
CFE_TIME_SET_TIME_CC
 default_cfe_time_fncodes.h, 1663
CFE_TIME_SetLeapSecondsCmd, 802
 CommandHeader, 802
 Payload, 802
CFE_TIME_SetLeapSecondsCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetMETCmd, 803
 CommandHeader, 803
 Payload, 803
CFE_TIME_SetMETCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetSignalCmd, 803
 CommandHeader, 803
 Payload, 804
CFE_TIME_SetSignalCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetSourceCmd, 804
 CommandHeader, 804
 Payload, 804
CFE_TIME_SetSourceCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetState
 default_cfe_time_extern_typedefs.h, 1651
CFE_TIME_SetState_Enum_t
 default_cfe_time_extern_typedefs.h, 1649
CFE_TIME_SetState_NOT_SET
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_SetState_WAS_SET
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_SetStateCmd, 804
 CommandHeader, 805
 Payload, 805
CFE_TIME_SetStateCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetSTCFCmd, 805
 CommandHeader, 805
 Payload, 805
CFE_TIME_SetSTCFCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SetTimeCmd, 806
 CommandHeader, 806
 Payload, 806
CFE_TIME_SetTimeCmd_t
 default_cfe_time_msgstruct.h, 1684
CFE_TIME_SIGNAL_CFG_EID
 cfe_time_eventids.h, 1694
CFE_TIME_SIGNAL_EID
 cfe_time_eventids.h, 1694
CFE_TIME_SIGNAL_ERR_EID
 cfe_time_eventids.h, 1694
CFE_TIME_SignalCmd_Payload, 806
 ToneSource, 807
CFE_TIME_SignalCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_SOURCE_CFG_EID
 cfe_time_eventids.h, 1694
CFE_TIME_SOURCE_EID
 cfe_time_eventids.h, 1695

CFE_TIME_SOURCE_ERR_EID
 cfe_time_eventids.h, 1695
CFE_TIME_SourceCmd_Payload, 807
 TimeSource, 807
CFE_TIME_SourceCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_SourceSelect
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_SourceSelect_Enum_t
 default_cfe_time_extern_typedefs.h, 1650
CFE_TIME_SourceSelect_EXTERNAL
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_SourceSelect_INTERNAL
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_STATE_EID
 cfe_time_eventids.h, 1695
CFE_TIME_STATE_ERR_EID
 cfe_time_eventids.h, 1695
CFE_TIME_StateCmd_Payload, 807
 ClockState, 808
CFE_TIME_StateCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_STCF_CFG_EID
 cfe_time_eventids.h, 1696
CFE_TIME_STCF_EID
 cfe_time_eventids.h, 1696
CFE_TIME_STCF_ERR_EID
 cfe_time_eventids.h, 1696
CFE_TIME_Sub2MicroSecs
 cFE Time Conversion APIs, 406
CFE_TIME_SUB_ADJUST_CC
 default_cfe_time_fncodes.h, 1664
CFE_TIME_SUB_DELAY_CC
 default_cfe_time_fncodes.h, 1665
CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC
 default_cfe_time_fncodes.h, 1666
CFE_TIME_SubAdjustCmd, 808
 CommandHeader, 808
 Payload, 808
CFE_TIME_SubAdjustCmd_t
 default_cfe_time_msgstruct.h, 1685
CFE_TIME_SubDelayCmd, 808
 CommandHeader, 809
 Payload, 809
CFE_TIME_SubDelayCmd_t
 default_cfe_time_msgstruct.h, 1685
CFE_TIME_SubOneHzAdjustmentCmd, 809
 CommandHeader, 809
 Payload, 809
CFE_TIME_SubOneHzAdjustmentCmd_t
 default_cfe_time_msgstruct.h, 1685
CFE_TIME_Subtract
 cFE Time Arithmetic APIs, 403
CFE_TIME_SynchCallbackPtr_t
 cfe_time_api_typedefs.h, 1410
CFE_TIME_SysTime, 810
 Seconds, 810
 Subseconds, 810
CFE_TIME_SysTime_t
 default_cfe_time_extern_typedefs.h, 1650
CFE_TIME_TIME_CFG_EID
 cfe_time_eventids.h, 1696
CFE_TIME_TIME_EID
 cfe_time_eventids.h, 1697
CFE_TIME_TIME_ERR_EID
 cfe_time_eventids.h, 1697
CFE_TIME_TimeCmd_Payload, 810
 MicroSeconds, 811
 Seconds, 811
CFE_TIME_TimeCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_TONE_CMD_MID
 default_cfe_time_msgids.h, 1682
CFE_TIME_ToneDataCmd, 811
 CommandHeader, 811
 Payload, 811
CFE_TIME_ToneDataCmd_Payload, 812
 AtToneLeapSeconds, 812
 AtToneMET, 812
 AtToneState, 812
 AtToneSTCF, 812
CFE_TIME_ToneDataCmd_Payload_t
 default_cfe_time_msgdefs.h, 1680
CFE_TIME_ToneDataCmd_t
 default_cfe_time_msgstruct.h, 1685
CFE_TIME_ToneSignalCmd, 812
 CommandHeader, 813
CFE_TIME_ToneSignalCmd_t
 default_cfe_time_msgstruct.h, 1685
CFE_TIME_ToneSignalSelect
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_ToneSignalSelect_Enum_t
 default_cfe_time_extern_typedefs.h, 1650
CFE_TIME_ToneSignalSelect_PRIMARY
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_ToneSignalSelect_REDUNDANT
 default_cfe_time_extern_typedefs.h, 1652
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS
 cFE Return Code Defines, 255
CFE_TIME_UnregisterSynchCallback
 cFE External Time Source APIs, 410
CFE_TIME_ZERO_VALUE
 cfe_time_api_typedefs.h, 1409
CFE_TST
 cfe_sb.h, 1399
cfe_version.h
 CFE_BUILD_BASELINE, 1411
 CFE_BUILD_CODENAME, 1411

CFE_BUILD_DEV_CYCLE, 1411
CFE_BUILD_NUMBER, 1412
CFE_CFG_MAX_VERSION_STR_LEN, 1412
CFE_LAST_OFFICIAL, 1412
CFE_MAJOR_VERSION, 1412
CFE_MINOR_VERSION, 1412
CFE_MISSION_REV, 1412
CFE_REVISION, 1412
CFE_SRC_VERSION, 1412
CFE_STR, 1412
CFE_STR_HELPER, 1413
CFECoreChecksum
 CFE_ES_HousekeepingTlm_Payload, 675
CFEMajorVersion
 CFE_ES_HousekeepingTlm_Payload, 675
CFEMinorVersion
 CFE_ES_HousekeepingTlm_Payload, 675
CFEMissionRevision
 CFE_ES_HousekeepingTlm_Payload, 675
CFERevision
 CFE_ES_HousekeepingTlm_Payload, 675
CFS CFDP Command Codes, 195
 CF_ABANDON_CC, 205
 CF_CANCEL_CC, 204
 CF_CMDS, 195
 CF_DISABLE_DEQUEUE_CC, 210
 CF_DISABLE_DIR_POLLING_CC, 212
 CF_DISABLE_ENGINE_CC, 215
 CF_ENABLE_DEQUEUE_CC, 209
 CF_ENABLE_DIR_POLLING_CC, 211
 CF_ENABLE_ENGINE_CC, 214
 CF_FREEZE_CC, 200
 CF_GET_PARAM_CC, 207
 CF_NOOP_CC, 196
 CF_NUM_COMMANDS, 215
 CF_PLAYBACK_DIR_CC, 199
 CF_PURGE_QUEUE_CC, 213
 CF_RESET_CC, 197
 CF_RESUME_CC, 203
 CF_SET_PARAM_CC, 206
 CF_SUSPEND_CC, 202
 CF_THAW_CC, 201
 CF_TX_FILE_CC, 198
 CF_WRITE_QUEUE_CC, 208
CFS CFDP Command Message IDs, 231
 CF_CMD_MID, 231
 CF_SEND_HK_MID, 231
 CF_WAKE_UP_MID, 231
CFS CFDP Command Structures, 224
 CF_AbandonCmd_t, 226
 CF_CancelCmd_t, 227
 CF_DisableDequeueCmd_t, 227
 CF_DisableDirPollingCmd_t, 227
 CF_DisableEngineCmd_t, 227
 CF_EnableDequeueCmd_t, 227
 CF_EnableDirPollingCmd_t, 227
 CF_EnableEngineCmd_t, 227
 CF_FreezeCmd_t, 227
 CF_GetParam_Payload_t, 227
 CF_GetParamCmd_t, 227
 CF_GetSet_ValueID_ack_limit, 229
 CF_GetSet_ValueID_ack_timer_s, 229
 CF_GetSet_ValueID_chan_max_outgoing_messages_per_wakeup,
 229
 CF_GetSet_ValueID_inactivity_timer_s, 229
 CF_GetSet_ValueID_local_eid, 229
 CF_GetSet_ValueID_MAX, 229
 CF_GetSet_ValueID_nak_limit, 229
 CF_GetSet_ValueID_nak_timer_s, 229
 CF_GetSet_ValueID_outgoing_file_chunk_size, 229
 CF_GetSet_ValueID_rx_crc_calc_bytes_per_wakeup,
 229
 CF_GetSet_ValueID_t, 229
 CF_GetSet_ValueID_ticks_per_second, 229
 CF_NoopCmd_t, 227
 CF_PlaybackDirCmd_t, 227
 CF_PurgeQueueCmd_t, 227
 CF_Queue_active, 229
 CF_Queue_all, 229
 CF_Queue_history, 229
 CF_Queue_pend, 229
 CF_Queue_t, 229
 CF_Reset_all, 230
 CF_Reset_command, 230
 CF_Reset_down, 230
 CF_Reset_fault, 230
 CF_Reset_t, 229
 CF_Reset_up, 230
 CF_ResetCountersCmd_t, 228
 CF_ResumeCmd_t, 228
 CF_SendHkCmd_t, 228
 CF_SetParam_Payload_t, 228
 CF_SetParamCmd_t, 228
 CF_SuspendCmd_t, 228
 CF_ThawCmd_t, 228
 CF_Transaction_Payload_t, 228
 CF_TxFile_Payload_t, 228
 CF_TxFileCmd_t, 228
 CF_Type_all, 230
 CF_Type_down, 230
 CF_Type_t, 230
 CF_Type_up, 230
 CF_UnionArgs_Payload_t, 228
 CF_WakeupCmd_t, 228
 CF_WriteQueue_Payload_t, 228
 CF_WriteQueueCmd_t, 229
CFS CFDP Data Interface Message IDs, 233
 CF_CH0_RX_MID, 233

CF_CH0_TX_MID, 233
 CF_CH1_RX_MID, 233
 CF_CH1_TX_MID, 233
CFS CFDP Event IDs, 152
 CF_CC_ERR_EID, 158
 CF_CFDP_CLOSE_ERR_EID, 158
 CF_CFDP_DIR_SLOT_ERR_EID, 158
 CF_CFDP_FD_UNHANDLED_ERR_EID, 159
 CF_CFDP_IDLE_MD_ERR_EID, 159
 CF_CFDP_INVALID_DST_ERR_EID, 159
 CF_CFDP_MAX_CMD_TX_ERR_EID, 159
 CF_CFDP_NO_MSG_ERR_EID, 160
 CF_CFDP_OPENDIR_ERR_EID, 160
 CF_CFDP_R_ACK_LIMIT_ERR_EID, 160
 CF_CFDP_R_CRC_ERR_EID, 160
 CF_CFDP_R_CREAT_ERR_EID, 161
 CF_CFDP_R_DC_INV_ERR_EID, 161
 CF_CFDP_R_EOF_MD_SIZE_ERR_EID, 161
 CF_CFDP_R_INACT_TIMER_ERR_EID, 161
 CF_CFDP_R_NAK_LIMIT_ERR_EID, 162
 CF_CFDP_R_OPEN_ERR_EID, 162
 CF_CFDP_R_PDU_EOF_ERR_EID, 162
 CF_CFDP_R_PDU_FINACK_ERR_EID, 162
 CF_CFDP_R_PDU_MD_ERR_EID, 163
 CF_CFDP_R_READ_ERR_EID, 163
 CF_CFDP_R_RENAME_ERR_EID, 163
 CF_CFDP_R_REQUEST_MD_INF_EID, 163
 CF_CFDP_R_SEEK_CRC_ERR_EID, 164
 CF_CFDP_R_SEEK_FD_ERR_EID, 164
 CF_CFDP_R_SIZE_MISMATCH_ERR_EID, 164
 CF_CFDP_R_TEMP_FILE_INF_EID, 164
 CF_CFDP_R_WRITE_ERR_EID, 165
 CF_CFDP_RX_DROPPED_ERR_EID, 165
 CF_CFDP_S_ACK_LIMIT_ERR_EID, 165
 CF_CFDP_S_ALREADY_OPEN_ERR_EID, 165
 CF_CFDP_S_DC_INV_ERR_EID, 166
 CF_CFDP_S_EARLY_FIN_ERR_EID, 166
 CF_CFDP_S_INACT_TIMER_ERR_EID, 166
 CF_CFDP_S_INVALID_SR_ERR_EID, 166
 CF_CFDP_S_NON_FD_PDU_ERR_EID, 167
 CF_CFDP_S_OPEN_ERR_EID, 167
 CF_CFDP_S_PDU_EOF_ERR_EID, 167
 CF_CFDP_S_PDU_NAK_ERR_EID, 167
 CF_CFDP_S_READ_ERR_EID, 168
 CF_CFDP_S_SEEK_BEG_ERR_EID, 168
 CF_CFDP_S_SEEK_END_ERR_EID, 168
 CF_CFDP_S_SEEK_FD_ERR_EID, 168
 CF_CFDP_S_SEND_FD_ERR_EID, 169
 CF_CFDP_S_SEND_MD_ERR_EID, 169
 CF_CFDP_S_START_SEND_INF_EID, 169
 CF_CMD_ABANDON_CHAN_ERR_EID, 169
 CF_CMD_ABANDON_INF_EID, 170
 CF_CMD_BAD_PARAM_ERR_EID, 170
 CF_CMD_CANCEL_CHAN_ERR_EID, 170

CF_CMD_CANCEL_INF_EID, 170
 CF_CMD_CHAN_PARAM_ERR_EID, 171
 CF_CMD_DISABLE_DEQUEUE_ERR_EID, 171
 CF_CMD_DISABLE_DEQUEUE_INF_EID, 171
 CF_CMD_DISABLE_ENGINE_INF_EID, 171
 CF_CMD_DISABLE_POLLDIR_ERR_EID, 172
 CF_CMD_DISABLE_POLLDIR_INF_EID, 172
 CF_CMD_ENABLE_DEQUEUE_ERR_EID, 172
 CF_CMD_ENABLE_DEQUEUE_INF_EID, 172
 CF_CMD_ENABLE_ENGINE_ERR_EID, 173
 CF_CMD_ENABLE_ENGINE_INF_EID, 173
 CF_CMD_ENABLE_POLLDIR_ERR_EID, 173
 CF_CMD_ENABLE_POLLDIR_INF_EID, 173
 CF_CMD_ENG_ALREADY_DIS_INF_EID, 174
 CF_CMD_ENG_ALREADY_ENA_INF_EID, 174
 CF_CMD_FREEZE_ERR_EID, 174
 CF_CMD_FREEZE_INF_EID, 174
 CF_CMD_GETSET1_INF_EID, 175
 CF_CMD_GETSET2_INF_EID, 175
 CF_CMD_GETSET_CHAN_ERR_EID, 175
 CF_CMD_GETSET_PARAM_ERR_EID, 175
 CF_CMD_GETSET_VALIDATE_ERR_EID, 176
 CF_CMD_LEN_ERR_EID, 176
 CF_CMD_PLAYBACK_DIR_ERR_EID, 176
 CF_CMD_PLAYBACK_DIR_INF_EID, 176
 CF_CMD_POLLDIR_INVALID_ERR_EID, 177
 CF_CMD_PURGE_ARG_ERR_EID, 177
 CF_CMD_PURGE_QUEUE_ERR_EID, 177
 CF_CMD_PURGE_QUEUE_INF_EID, 177
 CF_CMD_RESET_INVALID_ERR_EID, 178
 CF_CMD_SUSPRES_CHAN_ERR_EID, 178
 CF_CMD_SUSPRES_INF_EID, 178
 CF_CMD_SUSPRES SAME_INF_EID, 178
 CF_CMD_THAW_ERR_EID, 179
 CF_CMD_THAW_INF_EID, 179
 CF_CMD_TRANS_NOT_FOUND_ERR_EID, 179
 CF_CMD_TSN_CHAN_INVALID_ERR_EID, 179
 CF_CMD_TX_FILE_ERR_EID, 180
 CF_CMD_TX_FILE_INF_EID, 180
 CF_CMD_WHIST_WRITE_ERR_EID, 180
 CF_CMD_WQ_ARGS_ERR_EID, 180
 CF_CMD_WQ_CHAN_ERR_EID, 181
 CF_CMD_WQ_INF_EID, 181
 CF_CMD_WQ_OPEN_ERR_EID, 181
 CF_CMD_WQ_WRITEHIST_RX_ERR_EID, 181
 CF_CMD_WQ_WRITEHIST_TX_ERR_EID, 182
 CF_CMD_WQ_WRITEQ_PEND_ERR_EID, 182
 CF_CMD_WQ_WRITEQ_RX_ERR_EID, 182
 CF_CMD_WQ_WRITEQ_TX_ERR_EID, 182
 CF_CR_CHANNEL_PIPE_ERR_EID, 183
 CF_CR_PIPE_ERR_EID, 183
 CF_INIT_CRC_ALIGN_ERR_EID, 183
 CF_INIT_INF_EID, 183
 CF_INIT_MSG_RECV_ERR_EID, 184

CF_INIT_OUTGOING_SIZE_ERR_EID, 184
CF_INIT_SEM_ERR_EID, 184
CF_INIT_SUB_ERR_EID, 184
CF_INIT_TBL_CHECK_GA_ERR_EID, 185
CF_INIT_TBL_CHECK_MAN_ERR_EID, 185
CF_INIT_TBL_CHECK_REL_ERR_EID, 185
CF_INIT_TBL_GETADDR_ERR_EID, 185
CF_INIT_TBL_LOAD_ERR_EID, 186
CF_INIT_TBL_MANAGE_ERR_EID, 186
CF_INIT_TBL_REG_ERR_EID, 186
CF_INIT_TPS_ERR_EID, 186
CF_MID_ERR_EID, 187
CF_NOOP_INF_EID, 187
CF_PDU_ACK_SHORT_ERR_EID, 187
CF_PDU_EOF_SHORT_ERR_EID, 187
CF_PDU_FD_SHORT_ERR_EID, 188
CF_PDU_FD_UNSUPPORTED_ERR_EID, 188
CF_PDU_FIN_SHORT_ERR_EID, 188
CF_PDU_INVALID_DST_LEN_ERR_EID, 188
CF_PDU_INVALID_SRC_LEN_ERR_EID, 189
CF_PDU_LARGE_FILE_ERR_EID, 189
CF_PDU_MD_RECVD_INF_EID, 189
CF_PDU_MD_SHORT_ERR_EID, 189
CF_PDU_NAK_SHORT_ERR_EID, 190
CF_PDU_SHORT_HEADER_ERR_EID, 190
CF_PDU_TRUNCATION_ERR_EID, 190
CF_RESET_FREED_XACT_DBG_EID, 190
CF_RESET_INF_EID, 191
CFS CFDP Mission Configuration, 192
CF_PERF_ID_APPMAIN, 192
CF_PERF_ID_CREAT, 192
CF_PERF_ID_CYCLE_ENG, 192
CF_PERF_ID_DIRREAD, 192
CF_PERF_ID_FCLOSE, 193
CF_PERF_ID_FOPEN, 193
CF_PERF_ID_FREAD, 193
CF_PERF_ID_FSEEK, 193
CF_PERF_ID_FWRITE, 193
CF_PERF_ID_PDURCVD, 193
CF_PERF_ID_PDUSENT, 193
CF_PERF_ID_RENAME, 193
CFS CFDP Platform Configuration, 216
CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION, 217
CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION, 217
CF_CONFIG_TABLE_FILENAME, 217
CF_CONFIG_TABLE_NAME, 217
CF_FILENAME_MAX_LEN, 218
CF_FILENAME_MAX_NAME, 218
CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PERCHAN, 218
CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN, 218
CF_MAX_PDU_SIZE, 218
CF_MAX_POLLING_DIR_PER_CHAN, 219
CF_MAX_SIMULTANEOUS_RX, 219
CF_NAK_MAX_SEGMENTS, 219
CF_NUM_CHANNELS, 219
CF_NUM_HISTORIES_PER_CHANNEL, 220
CF_NUM_TRANSACTIONS_PER_PLAYBACK, 220
CF_PDU_ENCAPSULATION_EXTRA_TRAILING_BYTES, 220
CF_PIPE_DEPTH, 220
CF_R2_CRC_CHUNK_SIZE, 221
CF_RCVMMSG_TIMEOUT, 221
CF_STARTUP_SEM_MAX_RETRIES, 221
CF_STARTUP_SEM_TASK_DELAY, 221
CFS CFDP Telemetry, 222
CF_EotPacket_Payload_t, 223
CF_EotPacket_t, 223
CF_HkChannel_Data_t, 223
CF_HkCmdCounters_t, 223
CF_HkCounters_t, 223
CF_HkFault_t, 223
CF_HkPacket_Payload_t, 223
CF_HkPacket_t, 223
CF_HkRecv_t, 223
CF_HkSent_t, 223
CFS CFDP Telemetry Message IDs, 232
CF_EOT_TLM_MID, 232
CF_HK_TLM_MID, 232
CFS CFDP Version, 194
CF_MAJOR_VERSION, 194
CF_MINOR_VERSION, 194
CF_REVISION, 194
chan
CF_CFDP_CycleTx_args, 557
CF_CFDP_Tick_args, 566
CF_ConfigTable, 581
CF_Transaction_Payload, 647
CF_WriteQueue_Payload, 657
chan_num
CF_GetParam_Payload, 597
CF_SetParam_Payload, 638
CF_Transaction, 644
channel
CF_EotPacket_Payload, 591
channel_hk
CF_HkPacket_Payload, 606
channels
CF_Engine, 588
CheckErrCtr
CF_Logical_PduMd, 621
checksum_type
CF_MemPoolStats, 681
chunk_mem

CF_Engine, 589
 chunks
 CF_ChunkList, 578
 CF_ChunkWrapper, 579
 CF_Engine, 589
 CF_Transaction, 644
 cl_node
 CF_ChunkWrapper, 579
 CF_History, 599
 CF_Transaction, 644
 ClockFlyState
 CFE_TIME_DiagnosticTlm_Payload, 790
 ClockSetState
 CFE_TIME_DiagnosticTlm_Payload, 790
 ClockSignal
 CFE_TIME_DiagnosticTlm_Payload, 790
 ClockSource
 CFE_TIME_DiagnosticTlm_Payload, 790
 ClockState
 CFE_TIME_StateCmd_Payload, 808
 ClockStateAPI
 CFE_TIME_DiagnosticTlm_Payload, 790
 CFE_TIME_HousekeepingTlm_Payload, 797
 ClockStateFlags
 CFE_TIME_DiagnosticTlm_Payload, 790
 CFE_TIME_HousekeepingTlm_Payload, 797
 close_req
 CF_Logical_PduMd, 621
 cmd
 CF_HkCmdCounters, 602
 cmd_tx
 CF_Flags_Tx, 595
 CmdPipe
 CF_AppData_t, 555
 code_address
 OS_module_address_t, 819
 code_size
 OS_module_address_t, 819
 CodeAddress
 CFE_ES_AppInfo, 662
 codec_state
 CF_DecoderState, 583
 CF_EncoderState, 588
 CodeSize
 CFE_ES_AppInfo, 662
 com
 CF_Flags_Rx, 594
 CF_Flags_Tx, 595
 CF_StateFlags, 640
 CommandCounter
 CFE_ES_HousekeepingTlm_Payload, 675
 CFE_EVS_HousekeepingTlm_Payload, 717
 CFE_SB_HousekeepingTlm_Payload, 739
 CFE_TBL_HousekeepingTlm_Payload, 770
 CFE_TIME_HousekeepingTlm_Payload, 797
 CommandErrorCounter
 CFE_ES_HousekeepingTlm_Payload, 676
 CFE_EVS_HousekeepingTlm_Payload, 718
 CFE_SB_HousekeepingTlm_Payload, 739
 CFE_TBL_HousekeepingTlm_Payload, 770
 CFE_TIME_HousekeepingTlm_Payload, 797
 CommandHeader
 CF_AbandonCmd, 554
 CF_CancelCmd, 556
 CF_DisableDequeueCmd, 584
 CF_DisableDirPollingCmd, 585
 CF_DisableEngineCmd, 585
 CF_EnableDequeueCmd, 586
 CF_EnableDirPollingCmd, 586
 CF_EnableEngineCmd, 587
 CF_FreezeCmd, 596
 CF_GetParamCmd, 598
 CF_NoopCmd, 626
 CF_PlaybackDirCmd, 631
 CF_PurgeQueueCmd, 634
 CF_ResetCountersCmd, 634
 CF_ResumeCmd, 635
 CF_SendHkCmd, 637
 CF_SetParamCmd, 639
 CF_SuspendCmd, 641
 CF_ThawCmd, 642
 CF_TxFileCmd, 653
 CF_WakeupCmd, 657
 CF_WriteQueueCmd, 658
 CFE_ES_ClearERLogCmd, 668
 CFE_ES_ClearSysLogCmd, 669
 CFE_ES_DeleteCDSCmd, 669
 CFE_ES_DumpCDSRegistryCmd, 670
 CFE_ES_FileNameCmd, 671
 CFE_ES_NoopCmd, 683
 CFE_ES_OverWriteSysLogCmd, 684
 CFE_ES_QueryAllCmd, 687
 CFE_ES_QueryAllTasksCmd, 688
 CFE_ES_QueryOneCmd, 688
 CFE_ES_ReloadAppCmd, 689
 CFE_ES_ResetCountersCmd, 689
 CFE_ES_ResetPRCountCmd, 690
 CFE_ES_RestartAppCmd, 690
 CFE_ES_RestartCmd, 691
 CFE_ES_SendHkCmd, 692
 CFE_ES_SendMemPoolStatsCmd, 692
 CFE_ES_SetMaxPRCountCmd, 694
 CFE_ES_SetPerfFilterMaskCmd, 695
 CFE_ES_SetPerfTriggerMaskCmd, 696
 CFE_ES_StartApp, 697
 CFE_ES_StartPerfDataCmd, 699
 CFE_ES_StopAppCmd, 700
 CFE_ES_StopPerfDataCmd, 701

CFE_ES_WriteERLogCmd, 703
CFE_ES_WriteSysLogCmd, 704
CFE_EVS_AddEventFilterCmd, 704
CFE_EVS_ClearLogCmd, 710
CFE_EVS_DeleteEventFilterCmd, 711
CFE_EVS_DisableAppEventsCmd, 711
CFE_EVS_DisableAppEventTypeCmd, 712
CFE_EVS_DisableEventTypeCmd, 713
CFE_EVS_DisablePortsCmd, 713
CFE_EVS_EnableAppEventsCmd, 714
CFE_EVS_EnableAppEventTypeCmd, 714
CFE_EVS_EnableEventTypeCmd, 715
CFE_EVS_EnablePortsCmd, 716
CFE_EVS_NoopCmd, 722
CFE_EVS_ResetAllFiltersCmd, 724
CFE_EVS_ResetAppCounterCmd, 724
CFE_EVS_ResetCountersCmd, 725
CFE_EVS_ResetFilterCmd, 725
CFE_EVS_SendHkCmd, 726
CFE_EVS_SetEventFormatModeCmd, 727
CFE_EVS_SetFilterCmd, 727
CFE_EVS_SetLogModeCmd, 729
CFE_EVS_WriteAppDataFileCmd, 730
CFE_EVS_WriteLogDataFileCmd, 731
CFE_SB_DisableRouteCmd, 736
CFE_SB_DisableSubReportingCmd, 737
CFE_SB_EnableRouteCmd, 737
CFE_SB_EnableSubReportingCmd, 738
CFE_SB_NoopCmd, 744
CFE_SB_ResetCountersCmd, 748
CFE_SB_SendHkCmd, 750
CFE_SB_SendPrevSubsCmd, 751
CFE_SB_SendSbStatsCmd, 751
CFE_SB_WriteMapInfoCmd, 758
CFE_SB_WritePipeInfoCmd, 759
CFE_SB_WriteRoutingInfoCmd, 760
CFE_TBL_AbortLoadCmd, 760
CFE_TBL_ActivateCmd, 761
CFE_TBL_DeleteCDSCmd, 763
CFE_TBL_DumpCmd, 763
CFE_TBL_DumpRegistryCmd, 765
CFE_TBL_LoadCmd, 775
CFE_TBL_NoopCmd, 776
CFE_TBL_NotifyCmd, 776
CFE_TBL_ResetCountersCmd, 777
CFE_TBL_SendHkCmd, 778
CFE_TBL_SendRegistryCmd, 778
CFE_TBL_ValidateCmd, 783
CFE_TIME_AddAdjustCmd, 785
CFE_TIME_AddDelayCmd, 785
CFE_TIME_AddOneHzAdjustmentCmd, 786
CFE_TIME_FakeToneCmd, 795
CFE_TIME_NoopCmd, 800
CFE_TIME_OneHzCmd, 800
CFE_TIME_ResetCountersCmd, 801
CFE_TIME_SendDiagnosticCmd, 801
CFE_TIME_SendHkCmd, 802
CFE_TIME_SetLeapSecondsCmd, 802
CFE_TIME_SetMETCcmd, 803
CFE_TIME_SetSignalCmd, 803
CFE_TIME_SetSourceCmd, 804
CFE_TIME_SetStateCmd, 805
CFE_TIME_SetSTCFCmd, 805
CFE_TIME_SetTimeCmd, 806
CFE_TIME_SubAdjustCmd, 808
CFE_TIME_SubDelayCmd, 809
CFE_TIME_SubOneHzAdjustmentCmd, 809
CFE_TIME_ToneDataCmd, 811
CFE_TIME_ToneSignalCmd, 813
common_types.h
 EXTENSION, 1699
 CompileTimeAssert, 1699, 1701, 1702
 cpuaddr, 1699
 cpudiff, 1700
 cpusize, 1700
 int16, 1700
 int32, 1700
 int64, 1700
 int8, 1700
 intptr, 1700
 OS_ArgCallback_t, 1700
 OS_PRINTF, 1699
 OS_USED, 1699
 OSAL_BLOCKCOUNT_C, 1699
 osal_blockcount_t, 1700
 osal_id_t, 1700
 OSAL_INDEX_C, 1699
 osal_index_t, 1700
 OSAL_OBJTYPE_C, 1699
 osal_objtype_t, 1701
 OSAL_SIZE_C, 1699
 OSAL_STATUS_C, 1699
 osal_status_t, 1701
 uint16, 1701
 uint32, 1701
 uint64, 1701
 uint8, 1701
CompileTimeAssert
 common_types.h, 1699, 1701, 1702
complete
 CF_Flags_Rx, 594
config_handle
 CF_AppData_t, 555
config_table
 CF_AppData_t, 555
cont
 CF_CFDP_Tick_args, 566
container_of

cf_clist.h, 1092
 content_crc
 CF_Logical_PduBuffer, 614
 ContentType
 CFE_FS_Header, 733
 context
 CF_TraverseAll_Arg, 651
 continuation_state
 CF_Logical_PduFileDataHeader, 616
 count
 CF_ChunkList, 578
 counted
 CF_Playback, 630
 counter
 CF_Traverse_WriteHistoryFileArg, 649
 CF_Traverse_WriteTxnFileArg, 650
 CF_TraverseAll_Arg, 651
 counters
 CF_HkChannel_Data, 600
 CF_HkPacket_Payload, 607
 cpuaddr
 common_types.h, 1699
 cpudiff
 common_types.h, 1700
 cupsize
 common_types.h, 1700
 Crc
 CFE_TBL_Info, 773
 CFE_TBL_TblRegPacket_Payload, 781
 crc
 CF_CFDP_PduEof, 559
 CF_Logical_PduEof, 615
 CF_Transaction, 644
 crc_calc
 CF_Flags_Common, 593
 crc_flag
 CF_Logical_PduHeader, 619
 crc_mismatch
 CF_HkFault, 604
 crc_result
 CF_EotPacket_Payload, 591
 CreatePipeErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 739
 creator
 OS_bin_sem_prop_t, 813
 OS_condvar_prop_t, 814
 OS_count_sem_prop_t, 814
 OS_mut_sem_prop_t, 821
 OS_queue_prop_t, 821
 OS_socket_prop_t, 824
 OS_task_prop_t, 825
 OS_timebase_prop_t, 827
 OS_timer_prop_t, 828
 Critical
 CFE_TBL_Info, 773
 CFE_TBL_TblRegPacket_Payload, 781
 cs
 CF_Channel, 573
 cur
 CF_Channel, 573
 CurrentLatch
 CFE_TIME_DiagnosticTlm_Payload, 790
 CurrentMET
 CFE_TIME_DiagnosticTlm_Payload, 791
 CurrentQueueDepth
 CFE_SB_PipeDepthStats, 745
 CFE_SB_PipeInfoEntry, 746
 CurrentTAI
 CFE_TIME_DiagnosticTlm_Payload, 791
 CurrentUTC
 CFE_TIME_DiagnosticTlm_Payload, 791
 data
 CF_CFDP_PduFileDataContent, 560
 CF_ChAction_BoolMsgArg, 571
 CF_ChAction_MsgArg, 572
 CF_Logical_Tlv, 624
 data_address
 OS_module_address_t, 819
 data_encoded_length
 CF_Logical_PduHeader, 619
 data_len
 CF_Logical_PduFileDataHeader, 616
 data_ptr
 CF_Logical_Lv, 612
 CF_Logical_PduFileDataHeader, 616
 CF_Logical_TlvData, 625
 data_size
 OS_module_address_t, 819
 DataAddress
 CFE_ES_AppInfo, 662
 DataFileName
 CFE_ES_StopPerfCmd_Payload, 701
 DataSize
 CFE_ES_AppInfo, 663
 DataStoreStatus
 CFE_TIME_DiagnosticTlm_Payload, 791
 Datum
 CFE_Config_ValueEntry, 660
 dc
 CF_RxS2_Data, 636
 decode
 CF_Input, 610
 default_cf_extern_typedefs.h
 CF_CFDP_CLASS_1, 830
 CF_CFDP_CLASS_2, 830
 CF_CFDP_Class_t, 830
 CF_EntityId_t, 829

CF_QueueIdx_FREE, 831
CF_QueueIdx_HIST, 830
CF_QueueIdx_HIST_FREE, 830
CF_QueueIdx_NUM, 831
CF_QueueIdx_PEND, 830
CF_QueueIdx_RX, 830
CF_QueueIdx_t, 830
CF_QueueIdx_TXA, 830
CF_QueueIdx_TXW, 830
CF_TransactionSeq_t, 829
CF_TxnFilenames_t, 830
default_cf_mission_cfg.h
 CF_FILENAME_MAX_PATH, 833
default_cf_msg.h
 CF_ALL_CHANNELS, 834
 CF_ALL_POLLDIRS, 834
 CF_COMPOUND_KEY, 834
default_cf_platform_cfg.h
 CF_MISSION_REV, 839
 CF_TOTAL_CHUNKS, 839
default_cf_tbldefs.h
 CF_ChannelConfig_t, 840
 CF_PollDir_t, 840
default_cf_tblstruct.h
 CF_ConfigTable_t, 841
default_cf_topicids.h
 CFE_MISSION_CF_CH0_RX_TOPICID, 842
 CFE_MISSION_CF_CH0_TX_TOPICID, 842
 CFE_MISSION_CF_CH1_RX_TOPICID, 842
 CFE_MISSION_CF_CH1_TX_TOPICID, 842
 CFE_MISSION_CF_CMD_TOPICID, 842
 CFE_MISSION_CF_EOT_TLM_TOPICID, 842
 CFE_MISSION_CF_HK_TLM_TOPICID, 842
 CFE_MISSION_CF_SEND_HK_TOPICID, 842
 CFE_MISSION_CF_WAKE_UP_TOPICID, 842
default_cfe_core_api_base_msgids.h
 CFE_CPU1_CMD_MID_BASE, 1349
 CFE_CPU1_TLM_MID_BASE, 1349
 CFE_GLOBAL_CMD_MID_BASE, 1349
 CFE_GLOBAL_CMD_TOPICID_TO_MIDV, 1349
 CFE_GLOBAL_TLM_MID_BASE, 1349
 CFE_GLOBAL_TLM_TOPICID_TO_MIDV, 1349
 CFE_PLATFORM_CMD_TOPICID_TO_MIDV, 1350
 CFE_PLATFORM_TLM_TOPICID_TO_MIDV, 1350
default_cfe_core_api_interface_cfg.h
 CFE_MISSION_MAX_API_LEN, 1350
 CFE_MISSION_MAX_FILE_LEN, 1351
 CFE_MISSION_MAX_NUM_FILES, 1351
 CFE_MISSION_MAX_PATH_LEN, 1352
default_cfe_es_extern_typedefs.h
 CFE_ES_AppId_t, 1416
 CFE_ES_AppInfo_t, 1416
 CFE_ES_AppState, 1419
 CFE_ES_AppState_EARLY_INIT, 1419
 CFE_ES_AppState_Enum_t, 1416
 CFE_ES_AppState_LATE_INIT, 1419
 CFE_ES_AppState_MAX, 1419
 CFE_ES_AppState_RUNNING, 1419
 CFE_ES_AppState_STOPPED, 1419
 CFE_ES_AppState_UNDEFINED, 1419
 CFE_ES_AppState_WAITING, 1419
 CFE_ES_AppType, 1419
 CFE_ES_AppType_CORE, 1420
 CFE_ES_AppType_Enum_t, 1416
 CFE_ES_AppType_EXTERNAL, 1420
 CFE_ES_AppType_LIBRARY, 1420
 CFE_ES_BlockStats_t, 1416
 CFE_ES_CDSHandle_t, 1416
 CFE_ES_CDSRegDumpRec_t, 1416
 CFE_ES_CounterId_t, 1417
 CFE_ES_ExceptionAction, 1420
 CFE_ES_ExceptionAction_Enum_t, 1417
 CFE_ES_ExceptionAction_PROC_RESTART, 1420
 CFE_ES_ExceptionAction_RESTART_APP, 1420
 CFE_ES_LibId_t, 1417
 CFE_ES_LogEntryType, 1420
 CFE_ES_LogEntryType_APPLICATION, 1420
 CFE_ES_LogEntryType_CORE, 1420
 CFE_ES_LogEntryType_Enum_t, 1417
 CFE_ES_LogMode, 1420
 CFE_ES_LogMode_DISCARD, 1420
 CFE_ES_LogMode_Enum_t, 1417
 CFE_ES_LogMode_OVERWRITE, 1420
 CFE_ES_MEMADDRESS_C, 1415
 CFE_ES_MemAddress_t, 1417
 CFE_ES_MEMADDRESS_TO_PTR, 1415
 CFE_ES_MemHandle_t, 1418
 CFE_ES_MEMOFFSET_C, 1415
 CFE_ES_MemOffset_t, 1418
 CFE_ES_MEMOFFSET_TO_SIZE, 1415
 CFE_ES_MemPoolStats_t, 1418
 CFE_ES_RunStatus, 1420
 CFE_ES_RunStatus_APP_ERROR, 1421
 CFE_ES_RunStatus_APP_EXIT, 1421
 CFE_ES_RunStatus_APP_RUN, 1421
 CFE_ES_RunStatus_CORE_APP_INIT_ERROR, 1421
 CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, 1421
 CFE_ES_RunStatus_Enum_t, 1418
 CFE_ES_RunStatus_MAX, 1421
 CFE_ES_RunStatus_SYS_DELETE, 1421
 CFE_ES_RunStatus_SYS_EXCEPTION, 1421
 CFE_ES_RunStatus_SYS_RELOAD, 1421
 CFE_ES_RunStatus_SYS_RESTART, 1421
 CFE_ES_RunStatus_UNDEFINED, 1420
 CFE_ES_SystemState, 1421
 CFE_ES_SystemState_APPS_INIT, 1421

CFE_ES_SystemState_CORE_READY, 1421
 CFE_ES_SystemState_CORE_STARTUP, 1421
 CFE_ES_SystemState_EARLY_INIT, 1421
 CFE_ES_SystemState_Enum_t, 1418
 CFE_ES_SystemState_MAX, 1421
 CFE_ES_SystemState_OPERATIONAL, 1421
 CFE_ES_SystemState_SHUTDOWN, 1421
 CFE_ES_SystemState_UNDEFINED, 1421
 CFE_ES_TaskId_t, 1419
 CFE_ES_TaskInfo_t, 1419
 CFE_ES_TaskPriority_Atom_t, 1419
 default_cfe_es_fcncodes.h
 CFE_ES_CLEAR_ER_LOG_CC, 1422
 CFE_ES_CLEAR_SYS_LOG_CC, 1422
 CFE_ES_DELETE_CDS_CC, 1423
 CFE_ES_DUMP_CDS_REGISTRY_CC, 1424
 CFE_ES_NOOP_CC, 1425
 CFE_ES_OVER_WRITE_SYS_LOG_CC, 1426
 CFE_ES_QUERY_ALL_CC, 1426
 CFE_ES_QUERY_ALL_TASKS_CC, 1427
 CFE_ES_QUERY_ONE_CC, 1428
 CFE_ES_RELOAD_APP_CC, 1429
 CFE_ES_RESET_COUNTERS_CC, 1430
 CFE_ES_RESET_PR_COUNT_CC, 1431
 CFE_ES_RESTART_APP_CC, 1431
 CFE_ES_RESTART_CC, 1432
 CFE_ES_SEND_MEM_POOL_STATS_CC, 1433
 CFE_ES_SET_MAX_PR_COUNT_CC, 1434
 CFE_ES_SET_PERF_FILTER_MASK_CC, 1435
 CFE_ES_SET_PERF_TRIGGER_MASK_CC, 1436
 CFE_ES_START_APP_CC, 1436
 CFE_ES_START_PERF_DATA_CC, 1437
 CFE_ES_STOP_APP_CC, 1438
 CFE_ES_STOP_PERF_DATA_CC, 1439
 CFE_ES_WRITE_ER_LOG_CC, 1440
 CFE_ES_WRITE_SYS_LOG_CC, 1441
 default_cfe_es_interface_cfg.h
 CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN, 1443
 CFE_MISSION_ES_CDS_MAX_NAME_LENGTH, 1443
 CFE_MISSION_ES_CRC_16, 1443
 CFE_MISSION_ES_CRC_32, 1443
 CFE_MISSION_ES_CRC_8, 1444
 CFE_MISSION_ES_DEFAULT_CRC, 1444
 CFE_MISSION_ES_MAX_APPLICATIONS, 1444
 CFE_MISSION_ES_PERF_MAX_IDS, 1444
 CFE_MISSION_ES_POOL_MAX_BUCKETS, 1444
 default_cfe_es_internal_cfg.h
 CFE_PLATFORM_ES_APP_KILL_TIMEOUT, 1447
 CFE_PLATFORM_ES_APP_SCAN_RATE, 1447
 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, 1448
 CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, 1448
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01, 1448
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02, 1448
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03, 1448
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04, 1448
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14, 1449
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15, 1450
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16, 1450
 CFE_PLATFORM_ES_CDS_SIZE, 1450
 CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE, 1450
 CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE, 1450
 CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE, 1451
 CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME, 1451
 CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE, 1451
 CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE, 1452
 CFE_PLATFORM_ES_DEFAULT_STACK_SIZE, 1452
 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE, 1452
 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE, 1453
 CFE_PLATFORM_ES_ER_LOG_ENTRIES, 1453

CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE, 1453
CFE_PLATFORM_ES_MAX_APPLICATIONS, 1454
CFE_PLATFORM_ES_MAX_BLOCK_SIZE, 1454
CFE_PLATFORM_ES_MAX_GEN_COUNTERS, 1454
CFE_PLATFORM_ES_MAX_LIBRARIES, 1454
CFE_PLATFORM_ES_MAX_MEMORY_POOLS, 1455
CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS, 1455
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01, 1455
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09, 1456
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15, 1457
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16, 1457
CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN, 1457
CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING, 1458
CFE_PLATFORM_ES_NONVOL_STARTUP_FILE, 1458
CFE_PLATFORM_ES_OBJECT_TABLE_SIZE, 1458
CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY, 1458
CFE_PLATFORM_ES_PERF_CHILD_PRIORITY, 1459
CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE, 1459
CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE, 1459
CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS, 1460
CFE_PLATFORM_ES_PERF_FILTMASK_ALL, 1460
CFE_PLATFORM_ES_PERF_FILTMASK_INIT, 1460
CFE_PLATFORM_ES_PERF_FILTMASK_NONE, 1460
CFE_PLATFORM_ES_PERF_TRIGMASK_ALL, 1461
CFE_PLATFORM_ES_PERF_TRIGMASK_INIT, 1461
CFE_PLATFORM_ES_PERF_TRIGMASK_NONE, 1461
CFE_PLATFORM_ES_POOL_MAX_BUCKETS, 1461
CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING, 1462
CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS, 1462
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED, 1462
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE, 1463
CFE_PLATFORM_ES_START_TASK_PRIORITY, 1463
CFE_PLATFORM_ES_START_TASK_STACK_SIZE, 1463
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC, 1464
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC, 1464
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, 1464
CFE_PLATFORM_ES_USER_RESERVED_SIZE, 1465
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE, 1465
default_cfe_es_msgdefs.h
CFE_ES_AppNameCmd_Payload_t, 1468
CFE_ES_AppReloadCmd_Payload_t, 1468
CFE_ES_DeleteCDSCmd_Payload_t, 1468
CFE_ES_DumpCDSRegistryCmd_Payload_t, 1468
CFE_ES_FileNameCmd_Payload_t, 1468
CFE_ES_HousekeepingTlm_Payload_t, 1468
CFE_ES_OneAppTlm_Payload_t, 1468
CFE_ES_OverWriteSysLogCmd_Payload_t, 1469
CFE_ES_PerfMode, 1470
CFE_ES_PerfMode_Enum_t, 1469
CFE_ES_PerfTrigger_CENTER, 1470
CFE_ES_PerfTrigger_END, 1470
CFE_ES_PerfTrigger_START, 1470

CFE_ES_PoolStatsTlm_Payload_t, 1469
 CFE_ES_RestartCmd_Payload_t, 1469
 CFE_ES_SendMemPoolStatsCmd_Payload_t, 1469
 CFE_ES_SetMaxPRCountCmd_Payload_t, 1469
 CFE_ES_SetPerfFilterMaskCmd_Payload_t, 1469
 CFE_ES_SetPerfTrigMaskCmd_Payload_t, 1469
 CFE_ES_StartAppCmd_Payload_t, 1469
 CFE_ES_StartPerfCmd_Payload_t, 1469
 CFE_ES_StopPerfCmd_Payload_t, 1469
default_cfe_es_msgids.h
 CFE_ES_APP_TLM_MID, 1470
 CFE_ES_CMD_MID, 1470
 CFE_ES_HK_TLM_MID, 1471
 CFE_ES_MEMSTATS_TLM_MID, 1471
 CFE_ES_SEND_HK_MID, 1471
default_cfe_es_msgstruct.h
 CFE_ES_ClearERLogCmd_t, 1473
 CFE_ES_ClearSysLogCmd_t, 1473
 CFE_ES_DeleteCDSCmd_t, 1473
 CFE_ES_DumpCDSRegistryCmd_t, 1473
 CFE_ES_FileNameCmd_t, 1473
 CFE_ES_HousekeepingTlm_t, 1473
 CFE_ES_MemStatsTlm_t, 1473
 CFE_ES_NoopCmd_t, 1474
 CFE_ES_OneAppTlm_t, 1474
 CFE_ES_OverWriteSysLogCmd_t, 1474
 CFE_ES_QueryAllCmd_t, 1474
 CFE_ES_QueryAllTasksCmd_t, 1474
 CFE_ES_QueryOneCmd_t, 1474
 CFE_ES_ReloadAppCmd_t, 1474
 CFE_ES_ResetCountersCmd_t, 1474
 CFE_ES_ResetPRCountCmd_t, 1474
 CFE_ES_RestartAppCmd_t, 1474
 CFE_ES_RestartCmd_t, 1474
 CFE_ES_SendHkCmd_t, 1474
 CFE_ES_SendMemPoolStatsCmd_t, 1474
 CFE_ES_SetMaxPRCountCmd_t, 1474
 CFE_ES_SetPerfFilterMaskCmd_t, 1475
 CFE_ES_SetPerfTriggerMaskCmd_t, 1475
 CFE_ES_StartAppCmd_t, 1475
 CFE_ES_StartPerfDataCmd_t, 1475
 CFE_ES_StopAppCmd_t, 1475
 CFE_ES_StopPerfDataCmd_t, 1475
 CFE_ES_WriteERLogCmd_t, 1475
 CFE_ES_WriteSysLogCmd_t, 1475
default_cfe_es_topicids.h
 CFE_MISSION_ES_APP_TLM_TOPICID, 1476
 CFE_MISSION_ES_CMD_TOPICID, 1476
 CFE_MISSION_ES_HK_TLM_TOPICID, 1476
 CFE_MISSION_ES_MEMSTATS_TLM_TOPICID, 1476
 CFE_MISSION_ES_SEND_HK_TOPICID, 1476
default_cfe_evs_extern_typedefs.h
 CFE_EVS_EventFilter, 1503
 CFE_EVS_EventFilter_BINARY, 1503
 CFE_EVS_EventFilter_Enum_t, 1502
 CFE_EVS_EventOutput, 1504
 CFE_EVS_EventOutput_Enum_t, 1503
 CFE_EVS_EventOutput_PORT1, 1504
 CFE_EVS_EventOutput_PORT2, 1504
 CFE_EVS_EventOutput_PORT3, 1504
 CFE_EVS_EventOutput_PORT4, 1504
 CFE_EVS_EventType, 1504
 CFE_EVS_EventType_CRITICAL, 1504
 CFE_EVS_EventType_DEBUG, 1504
 CFE_EVS_EventType_Enum_t, 1503
 CFE_EVS_EventType_ERROR, 1504
 CFE_EVS_EventType_INFORMATION, 1504
 CFE_EVS_LogMode, 1504
 CFE_EVS_LogMode_DISCARD, 1504
 CFE_EVS_LogMode_Enum_t, 1503
 CFE_EVS_LogMode_OVERWRITE, 1504
 CFE_EVS_MsgFormat, 1504
 CFE_EVS_MsgFormat_Enum_t, 1503
 CFE_EVS_MsgFormat_LONG, 1504
 CFE_EVS_MsgFormat_SHORT, 1504
default_cfe_evs_fcncodes.h
 CFE_EVS_ADD_EVENT_FILTER_CC, 1505
 CFE_EVS_CLEAR_LOG_CC, 1506
 CFE_EVS_DELETE_EVENT_FILTER_CC, 1507
 CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, 1507
 CFE_EVS_DISABLE_APP_EVENTS_CC, 1508
 CFE_EVS_DISABLE_EVENT_TYPE_CC, 1509
 CFE_EVS_DISABLE_PORTS_CC, 1510
 CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, 1511
 CFE_EVS_ENABLE_APP_EVENTS_CC, 1512
 CFE_EVS_ENABLE_EVENT_TYPE_CC, 1513
 CFE_EVS_ENABLE_PORTS_CC, 1514
 CFE_EVS_NOOP_CC, 1515
 CFE_EVS_RESET_ALL_FILTERS_CC, 1515
 CFE_EVS_RESET_APP_COUNTER_CC, 1516
 CFE_EVS_RESET_COUNTERS_CC, 1517
 CFE_EVS_RESET_FILTER_CC, 1518
 CFE_EVS_SET_EVENT_FORMAT_MODE_CC, 1518
 CFE_EVS_SET_FILTER_CC, 1519
 CFE_EVS_SET_LOG_MODE_CC, 1520
 CFE_EVS_WRITE_APP_DATA_FILE_CC, 1521
 CFE_EVS_WRITE_LOG_DATA_FILE_CC, 1522
default_cfe_evs_interface_cfg.h
 CFE_MISSION_EVS_MAX_MESSAGE_LENGTH, 1523
default_cfe_evs_internal_cfg.h
 CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC, 1524
 CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE, 1524

CFE_PLATFORM_EVS_DEFAULT_LOG_FILE, [1525](#)
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE,
 [1525](#)
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE, CFE_EVS_EnableEventCmd_t, [1535](#)
 [1525](#)
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG,
 [1525](#)
CFE_PLATFORM_EVS_LOG_MAX, [1526](#)
CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST,
 [1526](#)
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS,
 [1526](#)
CFE_PLATFORM_EVS_PORT_DEFAULT, [1527](#)
CFE_PLATFORM_EVS_START_TASK_PRIORITY,
 [1527](#)
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE,
 [1527](#)
default_cfe_evs_msgdefs.h
 CFE_EVS_AppDataCmd_Payload_t, [1530](#)
 CFE_EVS_AppNameBitMaskCmd_Payload_t, [1530](#)
 CFE_EVS_AppNameCmd_Payload_t, [1530](#)
 CFE_EVS_AppNameEventIDCmd_Payload_t, [1531](#)
 CFE_EVS_AppNameEventIDMaskCmd_Payload_t,
 [1531](#)
 CFE_EVS_AppTlmData_t, [1531](#)
 CFE_EVS_BitMaskCmd_Payload_t, [1531](#)
 CFE_EVS_CRITICAL_BIT, [1530](#)
 CFE_EVS_DEBUG_BIT, [1530](#)
 CFE_EVS_ERROR_BIT, [1530](#)
 CFE_EVS_HousekeepingTlm_Payload_t, [1531](#)
 CFE_EVS_INFORMATION_BIT, [1530](#)
 CFE_EVS_LogFileCmd_Payload_t, [1531](#)
 CFE_EVS_LongEventTlm_Payload_t, [1531](#)
 CFE_EVS_PacketID_t, [1531](#)
 CFE_EVS_PORT1_BIT, [1530](#)
 CFE_EVS_PORT2_BIT, [1530](#)
 CFE_EVS_PORT3_BIT, [1530](#)
 CFE_EVS_PORT4_BIT, [1530](#)
 CFE_EVS_SetEventFormatMode_Payload_t, [1531](#)
 CFE_EVS_SetLogMode_Payload_t, [1531](#)
 CFE_EVS_ShortEventTlm_Payload_t, [1531](#)
default_cfe_evs_msgids.h
 CFE_EVS_CMD_MID, [1532](#)
 CFE_EVS_HK_TLM_MID, [1532](#)
 CFE_EVS_LONG_EVENT_MSG_MID, [1532](#)
 CFE_EVS_SEND_HK_MID, [1532](#)
 CFE_EVS_SHORT_EVENT_MSG_MID, [1532](#)
default_cfe_evs_msgstruct.h
 CFE_EVS_AddEventFilterCmd_t, [1534](#)
 CFE_EVS_ClearLogCmd_t, [1534](#)
 CFE_EVS_DeleteEventFilterCmd_t, [1534](#)
 CFE_EVS_DisableAppEventsCmd_t, [1534](#)
 CFE_EVS_DisableAppEventTypeCmd_t, [1534](#)
 CFE_EVS_DisableEventTypeCmd_t, [1534](#)
CFE_EVS_DisablePortsCmd_t, [1534](#)
CFE_EVS_EnableAppEventsCmd_t, [1534](#)
CFE_EVS_EnableAppEventTypeCmd_t, [1535](#)
CFE_EVS_EnablePortsCmd_t, [1535](#)
CFE_EVS_HousekeepingTlm_t, [1535](#)
CFE_EVS_LongEventTlm_t, [1535](#)
CFE_EVS_NoopCmd_t, [1535](#)
CFE_EVS_ResetAllFiltersCmd_t, [1535](#)
CFE_EVS_ResetAppCounterCmd_t, [1535](#)
CFE_EVS_ResetCountersCmd_t, [1535](#)
CFE_EVS_ResetFilterCmd_t, [1535](#)
CFE_EVS_SendHkCmd_t, [1535](#)
CFE_EVS_SetEventFormatModeCmd_t, [1535](#)
CFE_EVS_SetFilterCmd_t, [1535](#)
CFE_EVS_SetLogModeCmd_t, [1535](#)
CFE_EVS_ShortEventTlm_t, [1535](#)
CFE_EVS_WriteAppDataFileCmd_t, [1535](#)
CFE_EVS_WriteLogFileCmd_t, [1536](#)
default_cfe_evs_topicids.h
 CFE_MISSION_EVS_CMD_TOPICID, [1536](#)
 CFE_MISSION_EVS_HK_TLM_TOPICID, [1536](#)
 CFE_MISSION_EVS_LONG_EVENT_MSG_TOPICID,
 [1537](#)
 CFE_MISSION_EVS_SEND_HK_TOPICID, [1537](#)
 CFE_MISSION_EVS_SHORT_EVENT_MSG_TOPICID,
 [1537](#)
default_cfe_fs_filedef.h
 CFE_FS_Header_t, [1550](#)
 CFE_FS_SubType, [1550](#)
 CFE_FS_SubType_Enum_t, [1550](#)
 CFE_FS_SubType_ES_CDS_REG, [1551](#)
 CFE_FS_SubType_ES_ERLOG, [1550](#)
 CFE_FS_SubType_ES_PERFDATA, [1551](#)
 CFE_FS_SubType_ES_QUERYALL, [1550](#)
 CFE_FS_SubType_ES_QUERYALLTASKS, [1551](#)
 CFE_FS_SubType_ES_SYSLOG, [1550](#)
 CFE_FS_SubType_EVS_APPDATA, [1551](#)
 CFE_FS_SubType_EVS_EVENTLOG, [1551](#)
 CFE_FS_SubType_SB_MAPDATA, [1551](#)
 CFE_FS_SubType_SB_PIPEDATA, [1551](#)
 CFE_FS_SubType_SB_ROUTEDATA, [1551](#)
 CFE_FS_SubType_TBL_IMG, [1551](#)
 CFE_FS_SubType_TBL_REG, [1551](#)
default_cfe_fs_interface_cfg.h
 CFE_FS_FILE_CONTENT_ID, [1551](#)
 CFE_FS_HDR_DESC_MAX_LEN, [1552](#)
default_cfe_sb_extern_typedefs.h
 CFE_SB_MsgId_Atom_t, [1555](#)
 CFE_SB_Pipeld_t, [1555](#)
 CFE_SB_QosPriority, [1556](#)
 CFE_SB_QosPriority_Enum_t, [1555](#)
 CFE_SB_QosPriority_HIGH, [1556](#)
 CFE_SB_QosPriority_LOW, [1556](#)

CFE_SB_QosReliability, [1556](#)
 CFE_SB_QosReliability_Enum_t, [1556](#)
 CFE_SB_QosReliability_HIGH, [1556](#)
 CFE_SB_QosReliability_LOW, [1556](#)
 CFE_SB_Routeld_Atom_t, [1556](#)
 CFE_SB_SUB_ENTRIES_PER_PKT, [1555](#)
 default_cfe_sb_fcncodes.h
 CFE_SB_DISABLE_ROUTE_CC, [1557](#)
 CFE_SB_DISABLE_SUB_REPORTING_CC, [1558](#)
 CFE_SB_ENABLE_ROUTE_CC, [1558](#)
 CFE_SB_ENABLE_SUB_REPORTING_CC, [1559](#)
 CFE_SB_NOOP_CC, [1560](#)
 CFE_SB_RESET_COUNTERS_CC, [1560](#)
 CFE_SB_SEND_PREV_SUBS_CC, [1561](#)
 CFE_SB_SEND_SB_STATS_CC, [1562](#)
 CFE_SB_WRITE_MAP_INFO_CC, [1563](#)
 CFE_SB_WRITE_PIPE_INFO_CC, [1564](#)
 CFE_SB_WRITE_ROUTING_INFO_CC, [1565](#)
 default_cfe_sb_interface_cfg.h
 CFE_MISSION_SB_MAX_PIPES, [1566](#)
 CFE_MISSION_SB_MAX_SB_MSG_SIZE, [1566](#)
 default_cfe_sb_internal_cfg.h
 CFE_PLATFORM_SB_BUF_MEMORY_BYTES, [1568](#)
 CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME, [1568](#)
 CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, [1569](#)
 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME, [1569](#)
 CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME, [1569](#)
 CFE_PLATFORM_SB_FILTER_MASK1, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK2, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK3, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK4, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK5, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK6, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK7, [1570](#)
 CFE_PLATFORM_SB_FILTER_MASK8, [1570](#)
 CFE_PLATFORM_SB_FILTERED_EVENT1, [1570](#)
 CFE_PLATFORM_SB_FILTERED_EVENT2, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT3, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT4, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT5, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT6, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT7, [1571](#)
 CFE_PLATFORM_SB_FILTERED_EVENT8, [1571](#)
 CFE_PLATFORM_SB_HIGHEST_VALID_MSGID, [1571](#)
 CFE_PLATFORM_SB_MAX_BLOCK_SIZE, [1572](#)
 CFE_PLATFORM_SB_MAX_DEST_PER_PKT, [1572](#)
 CFE_PLATFORM_SB_MAX_MSG_IDS, [1572](#)
 CFE_PLATFORM_SB_MAX_PIPES, [1572](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07, [1573](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15, [1574](#)
 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16, [1574](#)
 CFE_PLATFORM_SB_START_TASK_PRIORITY, [1574](#)
 CFE_PLATFORM_SB_START_TASK_STACK_SIZE, [1575](#)
 default_cfe_sb_msgdefs.h
 CFE_SB_AllSubscriptionsTlm_Payload_t, [1577](#)
 CFE_SB_HousekeepingTlm_Payload_t, [1577](#)
 CFE_SB_MsgMapFileEntry_t, [1577](#)
 CFE_SB_PipeDepthStats_t, [1577](#)
 CFE_SB_PipeInfoEntry_t, [1577](#)
 CFE_SB_RouteCmd_Payload_t, [1577](#)
 CFE_SB_RoutingFileEntry_t, [1578](#)
 CFE_SB_SingleSubscriptionTlm_Payload_t, [1578](#)
 CFE_SB_StatsTlm_Payload_t, [1578](#)
 CFE_SB_SubEntries_t, [1578](#)
 CFE_SB_WriteFileInfoCmd_Payload_t, [1578](#)
 default_cfe_sb_msgids.h
 CFE_SB_ALLSUBS_TLM_MID, [1579](#)
 CFE_SB_CMD_MID, [1579](#)
 CFE_SB_HK_TLM_MID, [1579](#)
 CFE_SB_ONESUB_TLM_MID, [1579](#)
 CFE_SB_SEND_HK_MID, [1579](#)

CFE_SB_STATS_TLM_MID, 1579
CFE_SB_SUB_RPT_CTRL_MID, 1579
default_cfe_sb_msgstruct.h
 CFE_SB_AllSubscriptionsTlm_t, 1580
 CFE_SB_DisableRouteCmd_t, 1581
 CFE_SB_DisableSubReportingCmd_t, 1581
 CFE_SB_EnableRouteCmd_t, 1581
 CFE_SB_EnableSubReportingCmd_t, 1581
 CFE_SB_HousekeepingTlm_t, 1581
 CFE_SB_NoopCmd_t, 1581
 CFE_SB_ResetCountersCmd_t, 1581
 CFE_SB_SendHkCmd_t, 1581
 CFE_SB_SendPrevSubsCmd_t, 1581
 CFE_SB_SendSbStatsCmd_t, 1581
 CFE_SB_SingleSubscriptionTlm_t, 1581
 CFE_SB_StatsTlm_t, 1581
 CFE_SB_WriteMapInfoCmd_t, 1581
 CFE_SB_WritePipeInfoCmd_t, 1581
 CFE_SB_WriteRoutingInfoCmd_t, 1581
default_cfe_sb_topicids.h
 CFE_MISSION_SB_ALLSUBS_TLM_TOPICID,
 1582
 CFE_MISSION_SB_CMD_TOPICID, 1582
 CFE_MISSION_SB_HK_TLM_TOPICID, 1582
 CFE_MISSION_SB_ONESUB_TLM_TOPICID, 1583
 CFE_MISSION_SB_SEND_HK_TOPICID, 1583
 CFE_MISSION_SB_STATS_TLM_TOPICID, 1583
 CFE_MISSION_SB_SUB_RPT_CTRL_TOPICID,
 1583
default_cfe_tbl_extern_typedefs.h
 CFE_TBL_BufferSelect, 1603
 CFE_TBL_BufferSelect_ACTIVE, 1604
 CFE_TBL_BufferSelect_Enum_t, 1603
 CFE_TBL_BufferSelect_INACTIVE, 1604
 CFE_TBL_File_Hdr_t, 1603
default_cfe_tbl_fcncodes.h
 CFE_TBL_ABORT_LOAD_CC, 1604
 CFE_TBL_ACTIVATE_CC, 1605
 CFE_TBL_DELETE_CDS_CC, 1606
 CFE_TBL_DUMP_CC, 1607
 CFE_TBL_DUMP_REGISTRY_CC, 1607
 CFE_TBL_LOAD_CC, 1608
 CFE_TBL_NOOP_CC, 1609
 CFE_TBL_RESET_COUNTERS_CC, 1610
 CFE_TBL_SEND_REGISTRY_CC, 1611
 CFE_TBL_VALIDATE_CC, 1612
default_cfe_tbl_interface_cfg.h
 CFE_MISSION_TBL_MAX_FULL_NAME_LEN,
 1613
 CFE_MISSION_TBL_MAX_NAME_LENGTH, 1614
default_cfe_tbl_internal_cfg.h
 CFE_PLATFORM_TBL_BUF_MEMORY_BYTES,
 1615
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE,
 1615
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES,
 1615
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE,
 1616
CFE_PLATFORM_TBL_MAX_NUM_HANDLES,
 1616
CFE_PLATFORM_TBL_MAX_NUM_TABLES, 1616
CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS,
 1616
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS LOADS,
 1617
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE,
 1617
CFE_PLATFORM_TBL_START_TASK_PRIORITY,
 1617
CFE_PLATFORM_TBL_START_TASK_STACK_SIZE,
 1618
CFE_PLATFORM_TBL_U32FROM4CHARS, 1618
CFE_PLATFORM_TBL_VALID_PRID_1, 1618
CFE_PLATFORM_TBL_VALID_PRID_2, 1618
CFE_PLATFORM_TBL_VALID_PRID_3, 1619
CFE_PLATFORM_TBL_VALID_PRID_4, 1619
CFE_PLATFORM_TBL_VALID_PRID_COUNT, 1619
CFE_PLATFORM_TBL_VALID_SCID_1, 1619
CFE_PLATFORM_TBL_VALID_SCID_2, 1619
CFE_PLATFORM_TBL_VALID_SCID_COUNT, 1619
default_cfe_tbl_msgdefs.h
 CFE_TBL_AbortLoadCmd_Payload_t, 1622
 CFE_TBL_ActivateCmd_Payload_t, 1622
 CFE_TBL_DelCDSCmd_Payload_t, 1622
 CFE_TBL_DumpCmd_Payload_t, 1622
 CFE_TBL_DumpRegistryCmd_Payload_t, 1622
 CFE_TBL_HousekeepingTlm_Payload_t, 1622
 CFE_TBL_LoadCmd_Payload_t, 1622
 CFE_TBL_NotifyCmd_Payload_t, 1622
 CFE_TBL_SendRegistryCmd_Payload_t, 1623
 CFE_TBL_TblRegPacket_Payload_t, 1623
 CFE_TBL_ValidateCmd_Payload_t, 1623
default_cfe_tbl_msgids.h
 CFE_TBL_CMD_MID, 1623
 CFE_TBL_HK_TLM_MID, 1623
 CFE_TBL_REG_TLM_MID, 1624
 CFE_TBL_SEND_HK_MID, 1624
default_cfe_tbl_msgstruct.h
 CFE_TBL_AbortLoadCmd_t, 1625
 CFE_TBL_ActivateCmd_t, 1625
 CFE_TBL_DeleteCDSCmd_t, 1625
 CFE_TBL_DumpCmd_t, 1625
 CFE_TBL_DumpRegistryCmd_t, 1625
 CFE_TBL_HousekeepingTlm_t, 1625
 CFE_TBL_LoadCmd_t, 1625
 CFE_TBL_NoopCmd_t, 1626

CFE_TBL_NotifyCmd_t, 1626
 CFE_TBL_ResetCountersCmd_t, 1626
 CFE_TBL_SendHkCmd_t, 1626
 CFE_TBL_SendRegistryCmd_t, 1626
 CFE_TBL_TableRegistryTlm_t, 1626
 CFE_TBL_ValidateCmd_t, 1626
 default_cfe_tbl_topicids.h
 CFE_MISSION_TBL_CMD_TOPICID, 1627
 CFE_MISSION_TBL_HK_TLM_TOPICID, 1627
 CFE_MISSION_TBL_REG_TLM_TOPICID, 1627
 CFE_MISSION_TBL_SEND_HK_TOPICID, 1627
 default_cfe_time_extern_typedefs.h
 CFE_TIME_AdjustDirection, 1650
 CFE_TIME_AdjustDirection_ADD, 1650
 CFE_TIME_AdjustDirection_Enum_t, 1649
 CFE_TIME_AdjustDirection_SUBTRACT, 1650
 CFE_TIME_ClockState, 1650
 CFE_TIME_ClockState_Enum_t, 1649
 CFE_TIME_ClockState_FLYWHEEL, 1651
 CFE_TIME_ClockState_INVALID, 1651
 CFE_TIME_ClockState_VALID, 1651
 CFE_TIME_FlagBit, 1651
 CFE_TIME_FlagBit_ADD1HZ, 1651
 CFE_TIME_FlagBit_ADDADJ, 1651
 CFE_TIME_FlagBit_ADDTCL, 1651
 CFE_TIME_FlagBit_CLKSET, 1651
 CFE_TIME_FlagBit_CMDFLY, 1651
 CFE_TIME_FlagBit_Enum_t, 1649
 CFE_TIME_FlagBit_FLYING, 1651
 CFE_TIME_FlagBit_GDTONE, 1651
 CFE_TIME_FlagBit_SERVER, 1651
 CFE_TIME_FlagBit_SIGPRI, 1651
 CFE_TIME_FlagBit_SRCINT, 1651
 CFE_TIME_FlagBit_SRVFLY, 1651
 CFE_TIME_FlywheelState, 1651
 CFE_TIME_FlywheelState_Enum_t, 1649
 CFE_TIME_FlywheelState_IS_FLY, 1651
 CFE_TIME_FlywheelState_NO_FLY, 1651
 CFE_TIME_SetState, 1651
 CFE_TIME_SetState_Enum_t, 1649
 CFE_TIME_SetState_NOT_SET, 1652
 CFE_TIME_SetState_WAS_SET, 1652
 CFE_TIME_SourceSelect, 1652
 CFE_TIME_SourceSelect_Enum_t, 1650
 CFE_TIME_SourceSelect_EXTERNAL, 1652
 CFE_TIME_SourceSelect_INTERNAL, 1652
 CFE_TIME_SysTime_t, 1650
 CFE_TIME_ToneSignalSelect, 1652
 CFE_TIME_ToneSignalSelect_Enum_t, 1650
 CFE_TIME_ToneSignalSelect_PRIMARY, 1652
 CFE_TIME_ToneSignalSelect_REDUNDANT, 1652
 default_cfe_time_fcncodes.h
 CFE_TIME_ADD_ADJUST_CC, 1653
 CFE_TIME_ADD_DELAY_CC, 1653
 CFE_TIME_ADD_ONE_HZ_ADJUSTMENT_CC, 1654
 CFE_TIME_NOOP_CC, 1655
 CFE_TIME_RESET_COUNTERS_CC, 1656
 CFE_TIME_SEND_DIAGNOSTIC_CC, 1657
 CFE_TIME_SET_LEAP_SECONDS_CC, 1658
 CFE_TIME_SET_MET_CC, 1659
 CFE_TIME_SET_SIGNAL_CC, 1660
 CFE_TIME_SET_SOURCE_CC, 1660
 CFE_TIME_SET_STATE_CC, 1661
 CFE_TIME_SET_STCF_CC, 1663
 CFE_TIME_SET_TIME_CC, 1663
 CFE_TIME_SUB_ADJUST_CC, 1664
 CFE_TIME_SUB_DELAY_CC, 1665
 CFE_TIME_SUB_ONE_HZ_ADJUSTMENT_CC, 1666
 default_cfe_time_interface_cfg.h
 CFE_MISSION_TIME_AT_TONE_WAS, 1668
 CFE_MISSION_TIME_AT_TONE_WILL_BE, 1669
 CFE_MISSION_TIME_CFG_DEFAULT_TAI, 1669
 CFE_MISSION_TIME_CFG_DEFAULT_UTC, 1669
 CFE_MISSION_TIME_CFG_FAKE_TONE, 1669
 CFE_MISSION_TIME_DEF_DELAY_SECS, 1669
 CFE_MISSION_TIME_DEF_DELAY_SUBS, 1669
 CFE_MISSION_TIME_DEF_LEAPS, 1670
 CFE_MISSION_TIME_DEF_MET_SECS, 1670
 CFE_MISSION_TIME_DEF_MET_SUBS, 1670
 CFE_MISSION_TIME_DEF_STCF_SECS, 1670
 CFE_MISSION_TIME_DEF_STCF_SUBS, 1670
 CFE_MISSION_TIME_EPOCH_DAY, 1670
 CFE_MISSION_TIME_EPOCH_HOUR, 1670
 CFE_MISSION_TIME_EPOCH_MICROS, 1670
 CFE_MISSION_TIME_EPOCH_MINUTE, 1671
 CFE_MISSION_TIME_EPOCH_SECOND, 1671
 CFE_MISSION_TIME_EPOCH_YEAR, 1671
 CFE_MISSION_TIME_FS_FACTOR, 1671
 CFE_MISSION_TIME_MAX_ELAPSED, 1671
 CFE_MISSION_TIME_MIN_ELAPSED, 1671
 default_cfe_time_internal_cfg.h
 CFE_PLATFORM_TIME_CFG_CLIENT, 1673
 CFE_PLATFORM_TIME_CFG_LATCH_FLY, 1673
 CFE_PLATFORM_TIME_CFG_SERVER, 1673
 CFE_PLATFORM_TIME_CFG_SIGNAL, 1673
 CFE_PLATFORM_TIME_CFG_SOURCE, 1673
 CFE_PLATFORM_TIME_CFG_SRC_GPS, 1674
 CFE_PLATFORM_TIME_CFG_SRC_MET, 1674
 CFE_PLATFORM_TIME_CFG_SRC_TIME, 1674
 CFE_PLATFORM_TIME_CFG_START_FLY, 1674
 CFE_PLATFORM_TIME_CFG_TONE_LIMIT, 1675
 CFE_PLATFORM_TIME_CFG_VIRTUAL, 1675
 CFE_PLATFORM_TIME_MAX_DELTA_SECS, 1675
 CFE_PLATFORM_TIME_MAX_DELTA_SUBS, 1676
 CFE_PLATFORM_TIME_MAX_LOCAL_SECS, 1676
 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS, 1676

CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY,
1676
CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE, default_cfe_time_topicids.h
1676
CFE_PLATFORM_TIME_START_TASK_PRIORITY,
1676
CFE_PLATFORM_TIME_START_TASK_STACK_SIZE,
1677
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY,
1677
CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE,
1677
default_cfe_time_msgdefs.h
CFE_TIME_DiagnosticTlm_Payload_t, 1680
CFE_TIME_HousekeepingTlm_Payload_t, 1680
CFE_TIME_LeapsCmd_Payload_t, 1680
CFE_TIME_OneHzAdjustmentCmd_Payload_t, 1680
CFE_TIME_SignalCmd_Payload_t, 1680
CFE_TIME_SourceCmd_Payload_t, 1680
CFE_TIME_StateCmd_Payload_t, 1680
CFE_TIME_TimeCmd_Payload_t, 1680
CFE_TIME_ToneDataCmd_Payload_t, 1680
default_cfe_time_msgids.h
CFE_TIME_1HZ_CMD_MID, 1681
CFE_TIME_CMD_MID, 1681
CFE_TIME_DATA_CMD_MID, 1681
CFE_TIME_DIAG_TLM_MID, 1681
CFE_TIME_HK_TLM_MID, 1681
CFE_TIME_ONEHZ_CMD_MID, 1681
CFE_TIME_SEND_CMD_MID, 1682
CFE_TIME_SEND_HK_MID, 1682
CFE_TIME_TONE_CMD_MID, 1682
default_cfe_time_msgstruct.h
CFE_TIME_AddAdjustCmd_t, 1683
CFE_TIME_AddDelayCmd_t, 1683
CFE_TIME_AddOneHzAdjustmentCmd_t, 1683
CFE_TIME_DiagnosticTlm_t, 1684
CFE_TIME_FakeToneCmd_t, 1684
CFE_TIME_HousekeepingTlm_t, 1684
CFE_TIME_NoopCmd_t, 1684
CFE_TIME_OneHzCmd_t, 1684
CFE_TIME_ResetCountersCmd_t, 1684
CFE_TIME_SendDiagnosticCmd_t, 1684
CFE_TIME_SendHkCmd_t, 1684
CFE_TIME_SetLeapSecondsCmd_t, 1684
CFE_TIME_SetMETCCmd_t, 1684
CFE_TIME_SetSignalCmd_t, 1684
CFE_TIME_SetSourceCmd_t, 1684
CFE_TIME_SetStateCmd_t, 1684
CFE_TIME_SetSTCFCmd_t, 1684
CFE_TIME_SetTimeCmd_t, 1684
CFE_TIME_SubAdjustCmd_t, 1685
CFE_TIME_SubDelayCmd_t, 1685
CFE_TIME_SubOneHzAdjustmentCmd_t, 1685
CFE_TIME_ToneDataCmd_t, 1685
CFE_TIME_ToneSignalCmd_t, 1685
CFE_MISSION_TIME_CMD_TOPICID, 1686
CFE_MISSION_TIME_DATA_CMD_TOPICID, 1686
CFE_MISSION_TIME_DIAG_TLM_TOPICID, 1686
CFE_MISSION_TIME_HK_TLM_TOPICID, 1686
CFE_MISSION_TIME_ONEHZ_CMD_TOPICID,
1686
CFE_MISSION_TIME_SEND_CMD_TOPICID, 1686
CFE_MISSION_TIME_SEND_HK_TOPICID, 1687
CFE_MISSION_TIME_TONE_CMD_TOPICID, 1687
DelayDirection
CFE_TIME_DiagnosticTlm_Payload, 791
delivery_code
CF_Logical_PduFin, 618
dequeue_enabled
CF_ChannelConfig, 575
Description
CFE_FS_FileWriteMetaData, 732
CFE_FS_Header, 733
CFE_TBL_FileDef, 767
dest_eid
CF_PollDir, 632
dest_filename
CF_Logical_PduMd, 621
dest_id
CF_Playback, 630
CF_TxFile_Payload, 652
destination_eid
CF_Logical_PduHeader, 619
dir
CF_History, 599
dir_id
CF_Playback, 630
direction
CF_EotPacket_Payload, 591
CF_Logical_PduHeader, 619
directive_and_subtype_code
CF_CFDP_PduAck, 559
directive_code
CF_CFDP_PduFileDirectiveHeader, 561
CF_Logical_PduFileDirectiveHeader, 617
directory_read
CF_HkFault, 604
diopen
CF_Playback, 630
DoubleBuffered
CFE_TBL_Info, 774
CFE_TBL_TblRegPacket_Payload, 781
dropped
CF_HkRecv, 608
dst_dir
CF_PollDir, 633

dst_filename
 CF_TxFile_Payload, 652
 CF_TxnFilenames, 654

DumpFilename
 CFE_ES_DumpCDSRegistryCmd_Payload, 671
 CFE_TBL_DumpCmd_Payload, 764
 CFE_TBL_DumpRegistryCmd_Payload, 766

DumpOnly
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 781

DuplicateSubscriptionsCounter
 CFE_SB_HousekeepingTlm_Payload, 740

dword
 CF_UnionArgs_Payload, 656

early_exit
 CF_CFDP_Tick_args, 567

eid
 CF_Logical_TlvData, 625
 CF_Transaction_Payload, 647

eid_length
 CF_Logical_PduHeader, 619

eid_tsn_lengths
 CF_CFDP_PduHeader, 562

ElementPtr
 CFE_Config_ArrayValue, 659

enabled
 CF_Engine, 589
 CF_PollDir, 633

encode
 CF_Output, 627

engine
 CF_AppData_t, 555

Entries
 CFE_SB_AllSubscriptionsTlm_Payload, 735

Entry
 CFE_SB_AllSubscriptionsTlm_Payload, 735

entry_point
 OS_module_prop_t, 820

EntryPoint
 CFE_ES_AppInfo, 663

eof
 CF_Logical_IntHeader, 611

eof_cc
 CF_RxS2_Data, 636

eof_crc
 CF_RxS2_Data, 636

eof_recv
 CF_Flags_Rx, 594

eof_size
 CF_RxS2_Data, 636

ERLogEntries
 CFE_ES_HousekeepingTlm_Payload, 676

ERLogIndex
 CFE_ES_HousekeepingTlm_Payload, 676

err
 CF_HkCmdCounters, 602

error
 CF_HkRecv, 608
 CF_Traverse_WriteHistoryFileArg, 649
 CF_Traverse_WriteTxnFileArg, 650

EventID
 CFE_EVS_AppNameEventIDCmd_Payload, 707
 CFE_EVS_AppNameEventIDMaskCmd_Payload, 708
 CFE_EVS_BinFilter, 709
 CFE_EVS_PacketID, 723

EventType
 CFE_EVS_PacketID, 723

example_mission_cfg.h
 CFE_FS_FILE_CONTENT_ID, 1288
 CFE_FS_HDR_DESC_MAX_LEN, 1289
 CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN, 1289
 CFE_MISSION_ES_CDS_MAX_NAME_LENGTH, 1289
 CFE_MISSION_ES_CRC_16, 1289
 CFE_MISSION_ES_CRC_32, 1289
 CFE_MISSION_ES_CRC_8, 1289
 CFE_MISSION_ES_DEFAULT_CRC, 1290
 CFE_MISSION_ES_MAX_APPLICATIONS, 1290
 CFE_MISSION_ES_PERF_MAX_IDS, 1290
 CFE_MISSION_ES_POOL_MAX_BUCKETS, 1290
 CFE_MISSION_EVS_MAX_MESSAGE_LENGTH, 1291
 CFE_MISSION_MAX_API_LEN, 1291
 CFE_MISSION_MAX_FILE_LEN, 1291
 CFE_MISSION_MAX_NUM_FILES, 1292
 CFE_MISSION_MAX_PATH_LEN, 1292
 CFE_MISSION_SB_MAX_PIPES, 1293
 CFE_MISSION_SB_MAX_SB_MSG_SIZE, 1293
 CFE_MISSION_TBL_MAX_FULL_NAME_LEN, 1293
 CFE_MISSION_TBL_MAX_NAME_LENGTH, 1294
 CFE_MISSION_TIME_AT_TONE_WAS, 1294
 CFE_MISSION_TIME_AT_TONE_WILL_BE, 1295
 CFE_MISSION_TIME_CFG_DEFAULT_TAI, 1295
 CFE_MISSION_TIME_CFG_DEFAULT_UTC, 1295
 CFE_MISSION_TIME_CFG_FAKE_TONE, 1295
 CFE_MISSION_TIME_DEF_DELAY_SECS, 1295
 CFE_MISSION_TIME_DEF_DELAY_SUBS, 1295
 CFE_MISSION_TIME_DEF_LEAPS, 1296
 CFE_MISSION_TIME_DEF_MET_SECS, 1296
 CFE_MISSION_TIME_DEF_MET_SUBS, 1296
 CFE_MISSION_TIME_DEF_STCF_SECS, 1296
 CFE_MISSION_TIME_DEF_STCF_SUBS, 1296
 CFE_MISSION_TIME_EPOCH_DAY, 1296
 CFE_MISSION_TIME_EPOCH_HOUR, 1296

- CFE_MISSION_TIME_EPOCH_MICROS, 1296
CFE_MISSION_TIME_EPOCH_MINUTE, 1297
CFE_MISSION_TIME_EPOCH_SECOND, 1297
CFE_MISSION_TIME_EPOCH_YEAR, 1297
CFE_MISSION_TIME_FS_FACTOR, 1297
CFE_MISSION_TIME_MAX_ELAPSED, 1297
CFE_MISSION_TIME_MIN_ELAPSED, 1297
example_platform_cfg.h
 CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, 1302
 CFE_PLATFORM_ENDIAN, 1302
 CFE_PLATFORM_ES_APP_KILL_TIMEOUT, 1302
 CFE_PLATFORM_ES_APP_SCAN_RATE, 1303
 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, 1303
 CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, 1303
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06, 1304
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15, 1305
 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16, 1305
 CFE_PLATFORM_ES_CDS_SIZE, 1306
 CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE, 1306
 CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE, CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14, 1306
 CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE, 1306
 CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME, 1307
 CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE, 1307
 CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE, 1307
 CFE_PLATFORM_ES_DEFAULT_STACK_SIZE, 1308
 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE, 1308
 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE, 1308
 CFE_PLATFORM_ES_ER_LOG_ENTRIES, 1309
 CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE, 1309
 CFE_PLATFORM_ES_MAX_APPLICATIONS, 1309
 CFE_PLATFORM_ES_MAX_BLOCK_SIZE, 1310
 CFE_PLATFORM_ES_MAX_GEN_COUNTERS, 1310
 CFE_PLATFORM_ES_MAX_LIBRARIES, 1310
 CFE_PLATFORM_ES_MAX_MEMORY_POOLS, 1310
 CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS, 1311
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01, 1311
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02, 1311
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13, 1312
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14, 1312

CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15,
 1313
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16,
 1313
CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN,
 1313
CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING,
 1313
CFE_PLATFORM_ES_NONVOL_STARTUP_FILE,
 1313
CFE_PLATFORM_ES_OBJECT_TABLE_SIZE, 1314
CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY,
 1314
CFE_PLATFORM_ES_PERF_CHILD_PRIORITY,
 1314
CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE,
 1314
CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE,
 1315
CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS,
 1315
CFE_PLATFORM_ES_PERF_FILTMASK_ALL, 1315
CFE_PLATFORM_ES_PERF_FILTMASK_INIT,
 1315
CFE_PLATFORM_ES_PERF_FILTMASK_NONE,
 1316
CFE_PLATFORM_ES_PERF_TRIGMASK_ALL,
 1316
CFE_PLATFORM_ES_PERF_TRIGMASK_INIT,
 1316
CFE_PLATFORM_ES_PERF_TRIGMASK_NONE,
 1316
CFE_PLATFORM_ES_POOL_MAX_BUCKETS,
 1317
CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING,
 1317
CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS,
 1317
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED
 1318
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE,
 1318
CFE_PLATFORM_ES_START_TASK_PRIORITY,
 1318
CFE_PLATFORM_ES_START_TASK_STACK_SIZE,
 1319
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSE
 1319
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC,
 1319
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, 1320
CFE_PLATFORM_ES_USER_RESERVED_SIZE,
 1320
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE,
 1320
CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC,
 1321
CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE,
 1321
CFE_PLATFORM_EVS_DEFAULT_LOG_FILE, 1321
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE,
 1322
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE,
 1322
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG,
 1322
CFE_PLATFORM_EVS_LOG_MAX, 1323
CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST,
 1323
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS,
 1323
CFE_PLATFORM_EVS_PORT_DEFAULT, 1324
CFE_PLATFORM_EVS_START_TASK_PRIORITY,
 1324
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE,
 1324
CFE_PLATFORM_SB_BUF_MEMORY_BYTES,
 1325
CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME,
 1325
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, 1325
CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME,
 1326
CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME,
 1326
CFE_PLATFORM_SB_FILTER_MASK1, 1326
CFE_PLATFORM_SB_FILTER_MASK2, 1326
CFE_PLATFORM_SB_FILTER_MASK3, 1327
CFE_PLATFORM_SB_FILTER_MASK4, 1327
CFE_PLATFORM_SB_FILTER_MASK5, 1327
CFE_PLATFORM_SB_FILTER_MASK6, 1327
CFE_PLATFORM_SB_FILTER_MASK7, 1327
CFE_PLATFORM_SB_FILTER_MASK8, 1327
CFE_PLATFORM_SB_FILTERED_EVENT1, 1327
CFE_PLATFORM_SB_FILTERED_EVENT2, 1327
CFE_PLATFORM_SB_FILTERED_EVENT3, 1327
CFE_PLATFORM_SB_FILTERED_EVENT4, 1327
CFE_PLATFORM_SB_FILTERED_EVENT5, 1328
CFE_PLATFORM_SB_FILTERED_EVENT6, 1328
CFE_PLATFORM_SB_FILTERED_EVENT7, 1328
CFE_PLATFORM_SB_FILTERED_EVENT8, 1328
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID,
 1328
CFE_PLATFORM_SB_MAX_BLOCK_SIZE, 1328
CFE_PLATFORM_SB_MAX_DEST_PER_PKT,
 1328
CFE_PLATFORM_SB_MAX_MSG_IDS, 1329
CFE_PLATFORM_SB_MAX_PIPES, 1329

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01, 1329
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10, 1330
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11, 1331
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12, 1331
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13, 1331
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14, 1331
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15, 1331
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16, 1331
CFE_PLATFORM_SB_START_TASK_PRIORITY, 1331
CFE_PLATFORM_SB_START_TASK_STACK_SIZE, 1331
CFE_PLATFORM_TBL_BUF_MEMORY_BYTES, 1332
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE, 1332
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES, 1332
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE, 1333
CFE_PLATFORM_TBL_MAX_NUM_HANDLES, 1333
CFE_PLATFORM_TBL_MAX_NUM_TABLES, 1333
CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS, 1334
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS, 1334
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE, 1334
CFE_PLATFORM_TBL_START_TASK_PRIORITY, 1335
CFE_PLATFORM_TBL_START_TASK_STACK_SIZE, 1335
CFE_PLATFORM_TBL_U32FROM4CHARS, 1335
CFE_PLATFORM_TBL_VALID_PRID_1, 1336
CFE_PLATFORM_TBL_VALID_PRID_2, 1336
CFE_PLATFORM_TBL_VALID_PRID_3, 1336
CFE_PLATFORM_TBL_VALID_PRID_4, 1336
CFE_PLATFORM_TBL_VALID_PRID_COUNT, 1336
CFE_PLATFORM_TBL_VALID_SCID_1, 1336
CFE_PLATFORM_TBL_VALID_SCID_2, 1337
CFE_PLATFORM_TBL_VALID_SCID_COUNT, 1337
CFE_PLATFORM_TIME_CFG_CLIENT, 1337
CFE_PLATFORM_TIME_CFG_LATCH_FLY, 1337
CFE_PLATFORM_TIME_CFG_SERVER, 1337
CFE_PLATFORM_TIME_CFG_SIGNAL, 1338
CFE_PLATFORM_TIME_CFG_SOURCE, 1338
CFE_PLATFORM_TIME_CFG_SRC_GPS, 1338
CFE_PLATFORM_TIME_CFG_SRC_MET, 1339
CFE_PLATFORM_TIME_CFG_SRC_TIME, 1339
CFE_PLATFORM_TIME_CFG_START_FLY, 1339
CFE_PLATFORM_TIME_CFG_TONE_LIMIT, 1339
CFE_PLATFORM_TIME_CFG_VIRTUAL, 1339
CFE_PLATFORM_TIME_MAX_DELTA_SECS, 1340
CFE_PLATFORM_TIME_MAX_DELTA_SUBS, 1340
CFE_PLATFORM_TIME_MAX_LOCAL_SECS, 1340
CFE_PLATFORM_TIME_MAX_LOCAL_SUBS, 1341
CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY, 1341
CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE, 1341
CFE_PLATFORM_TIME_START_TASK_PRIORITY, 1341
CFE_PLATFORM_TIME_START_TASK_STACK_SIZE, 1341
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY, 1342
CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE, 1342
ExceptionAction
 CFE_ES_AppInfo, 663
 CFE_ES_StartAppCmd_Payload, 698
ExecutionCounter
 CFE_ES_AppInfo, 663
 CFE_ES_TaskInfo, 702
fail_dir
 CF_ConfigTable, 581
FailedValCounter
 CFE_TBL_HousekeepingTlm_Payload, 770
fault
 CF_HkCounters, 602
fd
 CF_Logical_IntHeader, 611

CF_Transaction, 644
 CF_Traverse_WriteHistoryFileArg, 650
 CF_Traverse_WriteTxnFileArg, 650
 fd_nak_sent
 CF_Flags_Rx, 594
 fdirective
 CF_CFDP_FileDirectiveDispatchTable_t, 558
 CF_Logical_PduBuffer, 614
 FGV
 cf_codec.c, 1165
 file_data_bytes
 CF_HkRecv, 608
 CF_HkSent, 609
 file_open
 CF_HkFault, 604
 file_read
 CF_HkFault, 604
 file_rename
 CF_HkFault, 604
 file_seek
 CF_HkFault, 604
 file_size_mismatch
 CF_HkFault, 605
 file_status
 CF_Logical_PduFin, 618
 file_write
 CF_HkFault, 605
 FileModeBits
 os_fstat_t, 818
 FileName
 CFE_ES_AppInfo, 663
 CFE_ES_FileNameCmd_Payload, 672
 CFE_FS_FileWriteMetaData, 732
 os_dirent_t, 815
 Filename
 CFE_SB_WriteFileInfoCmd_Payload, 758
 filename
 CF_WriteQueue_Payload, 657
 OS_module_prop_t, 820
 FileSize
 os_fstat_t, 818
 FileSubType
 CFE_FS_FileWriteMetaData, 732
 FileTime
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 782
 os_fstat_t, 818
 filter_dir
 CF_Traverse_WriteHistoryFileArg, 650
 FilterMask
 CFE_ES_SetPerfFilterMaskCmd_Payload, 695
 FilterMaskNum
 CFE_ES_SetPerfFilterMaskCmd_Payload, 695
 fin
 CF_Logical_IntHeader, 611
 fin_cc
 CF_TxS2_Data, 655
 flags
 CF_CFDP_PduFin, 562
 CF_CFDP_PduHeader, 562
 CF_Transaction, 645
 OS_module_address_t, 820
 fn
 CF_CFDP_Tick_args, 567
 CF_TraverseAll_Arg, 651
 fnames
 CF_EotPacket_Payload, 591
 CF_History, 599
 CF_Playback, 630
 foofs
 CF_Transaction, 645
 Forced2Fly
 CFE_TIME_DiagnosticTlm_Payload, 791
 free_blocks
 OS_heap_prop_t, 818
 free_bytes
 OS_heap_prop_t, 818
 FreeFds
 os_fsinfo_t, 817
 freerun_time
 OS_timebase_prop_t, 827
 FreeVolumes
 os_fsinfo_t, 817
 frozen
 CF_HkChannel_Data, 601
 fs
 CF_RxS2_Data, 636
 fsize
 CF_EotPacket_Payload, 591
 CF_Transaction, 645
 FSV
 cf_codec.c, 1165
 GetData
 CFE_FS_FileWriteMetaData, 732
 GetPipeIdByNameErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 740
 Handle
 CFE_ES_CDSRegDumpRec, 667
 hdr
 CF_PduCmdMsg, 628
 CF_PduTlmMsg, 629
 header_encoded_length
 CF_Logical_PduHeader, 620
 HeapBlocksFree
 CFE_ES_HousekeepingTlm_Payload, 676
 HeapBytesFree
 CFE_ES_HousekeepingTlm_Payload, 676

HeapMaxBlockSize
 CFE_ES_HousekeepingTlm_Payload, 676

histories
 CF_Engine, 589

history
 CF_Transaction, 645

hk
 CF_AppData_t, 556

host_module_id
 OS_module_prop_t, 820

hword
 CF_UnionArgs_Payload, 656

in
 CF_Engine, 589

InactiveBufferAddr
 CFE_TBL_TblRegPacket_Payload, 782

inactivity_fired
 CF_Flags_Rx, 594

inactivity_timer
 CF_HkFault, 605
 CF_Transaction, 646

inactivity_timer_s
 CF_ChannelConfig, 575

Index
 CFE_SB_MsgMapFileEntry, 743

index
 CF_Crc, 583

int16
 common_types.h, 1700

int32
 common_types.h, 1700

int64
 common_types.h, 1700

int8
 common_types.h, 1700

int_header
 CF_Logical_PduBuffer, 614

InternalErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 740

interval_sec
 CF_PollDir, 633

interval_time
 OS_timer_prop_t, 828

interval_timer
 CF_Poll, 631

intptr
 common_types.h, 1700

is_valid
 CF_CodecState, 580

IsPending
 CFE_FS_FileWriteMetaData, 732

IsValid
 OS_file_prop_t, 816

keep
 CF_Playback, 630
 CF_Transaction, 646
 CF_TxFile_Payload, 652

key
 CF_GetParam_Payload, 597
 CF_SetParam_Payload, 638

large_flag
 CF_Logical_PduHeader, 620

largest_free_block
 OS_heap_prop_t, 819

LastFileDumped
 CFE_TBL_HousekeepingTlm_Payload, 770

LastFileLoaded
 CFE_TBL_HousekeepingTlm_Payload, 771
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 782

LastTableLoaded
 CFE_TBL_HousekeepingTlm_Payload, 771

LastUpdatedTable
 CFE_TBL_HousekeepingTlm_Payload, 771

LastUpdateTime
 CFE_TBL_HousekeepingTlm_Payload, 771

LastValCrc
 CFE_TBL_HousekeepingTlm_Payload, 771

LastValStatus
 CFE_TBL_HousekeepingTlm_Payload, 771

LastValTableName
 CFE_TBL_HousekeepingTlm_Payload, 771

LeapSeconds
 CFE_TIME_HousekeepingTlm_Payload, 797
 CFE_TIME_LeapsCmd_Payload, 799

Length
 CCSDS_PrimaryHeader, 554
 CFE_FS_Header, 733

length
 CF_CFDP_lv, 558
 CF_CFDP_PduHeader, 562
 CF_CFDP_tlv, 567
 CF_Logical_Lv, 612
 CF_Logical_Tlv, 624

LENGTHCHECK
 osapi-macros.h, 1722

LoadFilename
 CFE_TBL_LoadCmd_Payload, 775

LoadPending
 CFE_TBL_TblRegPacket_Payload, 782

local_eid
 CF_ConfigTable, 582

LocallntCounter
 CFE_TIME_DiagnosticTlm_Payload, 791

LocalTaskCounter
 CFE_TIME_DiagnosticTlm_Payload, 792

LogEnabled
 CFE_EVS_HousekeepingTlm_Payload, 718

LogFilename
 CFE_EVS_LogFileCmd_Payload, 720

LogFullFlag
 CFE_EVS_HousekeepingTlm_Payload, 718

LogMode
 CFE_EVS_HousekeepingTlm_Payload, 718
 CFE_EVS_SetLogMode_Payload, 728

LogOverflowCounter
 CFE_EVS_HousekeepingTlm_Payload, 718

LongDouble
 CFE_ES_PoolAlign, 686
 CFE_SB_Msg, 742

LongInt
 CFE_ES_PoolAlign, 686
 CFE_SB_Msg, 742

MainTaskId
 CFE_ES_AppInfo, 663

MainTaskName
 CFE_ES_AppInfo, 663

Mask
 CFE_EVS_AppNameEventIDMaskCmd_Payload,
 708
 CFE_EVS_BinFilter, 709

mask
 CF_Codec_BitField, 580

max_chunks
 CF_ChunkList, 578

max_outgoing_messages_per_wakeup
 CF_ChannelConfig, 575

max_size
 CF_CodecState, 580

MaxElapsed
 CFE_TIME_DiagnosticTlm_Payload, 792

MaxFds
 os_fsinfo_t, 817

MaxLocalClock
 CFE_TIME_DiagnosticTlm_Payload, 792

MaxMemAllowed
 CFE_SB_StatsTlm_Payload, 754

MaxMsgIdsAllowed
 CFE_SB_StatsTlm_Payload, 754

MaxPipeDepthAllowed
 CFE_SB_StatsTlm_Payload, 755

MaxPipesAllowed
 CFE_SB_StatsTlm_Payload, 755

MaxPRCount
 CFE_ES_SetMaxPRCountCmd_Payload, 694

MaxProcessorResets
 CFE_ES_HousekeepingTlm_Payload, 676

MaxQueueDepth
 CFE_SB_PipeDepthStats, 745

CFE_SB_PipeInfoEntry, 746

MaxSubscriptionsAllowed
 CFE_SB_StatsTlm_Payload, 755

MaxVolumes
 os_fsinfo_t, 817

md
 CF_Logical_IntHeader, 611

md_need_send
 CF_Flags_Tx, 595

md_recv
 CF_Flags_Rx, 594

MemInUse
 CFE_SB_HousekeepingTlm_Payload, 740
 CFE_SB_StatsTlm_Payload, 755

MemPoolHandle
 CFE_SB_HousekeepingTlm_Payload, 740
 CFE_TBL_HousekeepingTlm_Payload, 772

Message
 CFE_EVS_LongEventTlm_Payload, 721

MessageFormatMode
 CFE_EVS_HousekeepingTlm_Payload, 718

MessageSendCounter
 CFE_EVS_HousekeepingTlm_Payload, 718

MessageTruncCounter
 CFE_EVS_HousekeepingTlm_Payload, 719

MicroSeconds
 CFE_TIME_TimeCmd_Payload, 811

mid_input
 CF_ChannelConfig, 575

mid_output
 CF_ChannelConfig, 576

MinElapsed
 CFE_TIME_DiagnosticTlm_Payload, 792

Mode
 CFE_ES_OverWriteSysLogCmd_Payload, 685

Module
 OS_static_symbol_record_t, 824

move_dir
 CF_ChannelConfig, 576

Msg
 CFE_SB_Msg, 742

msg
 CF_Input, 610
 CF_Output, 627

MsgCnt
 CFE_SB_RoutingFileEntry, 749

MsgFormat
 CFE_EVS_SetEventFormatCode_Payload, 726

MsgId
 CFE_SB_MsgMapFileEntry, 744
 CFE_SB_RouteCmd_Payload, 748
 CFE_SB_RoutingFileEntry, 750
 CFE_SB_SingleSubscriptionTlm_Payload, 752
 CFE_SB_SubEntries, 757

MsgIdsInUse
 CFE_SB_StatsTlm_Payload, 755

MsgLimitErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 740

MsgReceiveErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 740

MsgSendErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 741

nak
 CF_GapComputeArgs_t, 596
 CF_Logical_IntHeader, 611

nak_limit
 CF_ChannelConfig, 576
 CF_HkFault, 605

nak_segment_requests
 CF_HkRecv, 608
 CF_HkSent, 609

nak_timer_s
 CF_ChannelConfig, 576

Name
 CFE_Config_IdNameEntry, 659
 CFE_ES_AppInfo, 664
 CFE_ES_CDSRegDumpRec, 667
 CFE_TBL_TblRegPacket_Payload, 782
 OS_static_symbol_record_t, 824

name
 OS_bin_sem_prop_t, 813
 OS_condvar_prop_t, 814
 OS_count_sem_prop_t, 814
 OS_module_prop_t, 820
 OS_mut_sem_prop_t, 821
 OS_queue_prop_t, 822
 OS_socket_prop_t, 824
 OS_task_prop_t, 826
 OS_timebase_prop_t, 827
 OS_timer_prop_t, 828

next
 CF_CListNode, 579

next_offset
 CF_CodecState, 580

nominal_interval_time
 OS_timebase_prop_t, 827

NoSubscribersCounter
 CFE_SB_HousekeepingTlm_Payload, 741

num_cmd_tx
 CF_Channel, 573

num_segments
 CF_Logical_SegmentList, 623

num_tlv
 CF_Logical_TlvList, 626

num_ts
 CF_Playback, 630

NumBlocksRequested
 CFE_ES_MemPoolStats, 681

NumBytes
 CFE_TBL_File_Hdr, 766

NumCreated
 CFE_ES_BlockStats, 666

NumElements
 CFE_Config_ArrayValue, 659

NumFree
 CFE_ES_BlockStats, 666

NumFreeBytes
 CFE_ES_MemPoolStats, 682

NumFreeSharedBufs
 CFE_TBL_HousekeepingTlm_Payload, 772

NumLoadPending
 CFE_TBL_HousekeepingTlm_Payload, 772

NumOfChildTasks
 CFE_ES_AppInfo, 664

NumTables
 CFE_TBL_HousekeepingTlm_Payload, 772

NumUsers
 CFE_TBL_Info, 774

NumValRequests
 CFE_TBL_HousekeepingTlm_Payload, 772

object_ids
 OS_FdSet, 816

ObjectName
 CFE_TBL_FileDef, 767

ObjectSize
 CFE_TBL_FileDef, 768

octets
 CF_CFDP_uint16_t, 569
 CF_CFDP_uint32_t, 569
 CF_CFDP_uint64_t, 570
 CF_CFDP_uint8_t, 570

Offset
 CFE_TBL_File_Hdr, 766

offset
 CF_CFDP_PduFileDataHeader, 561
 CF_Chunk, 577
 CF_Logical_PduFileDataHeader, 616

offset_end
 CF_CFDP_SegmentRequest, 566
 CF_Logical_SegmentRequest, 624

offset_start
 CF_CFDP_SegmentRequest, 566
 CF_Logical_SegmentRequest, 624

OneHzAdjust
 CFE_TIME_DiagnosticTlm_Payload, 792

OneHzDirection
 CFE_TIME_DiagnosticTlm_Payload, 792

OneTimeAdjust
 CFE_TIME_DiagnosticTlm_Payload, 792

OneTimeDirection

CFE_TIME_DiagnosticTlm_Payload, 793
 OnEvent
 CFE_FS_FileWriteMetaData, 732
 OptS
 CFE_SB_PipeInfoEntry, 746
 OS_ADD_TASK_FLAGS
 osconfig.h, 1283
 OS_API_Init
 OSAL Core Operation APIs, 440
 OS_API_Teardown
 OSAL Core Operation APIs, 441
 OS_Application_Run
 OSAL Core Operation APIs, 441
 OS_Application_Startup
 OSAL Core Operation APIs, 441
 OS_ApplicationExit
 OSAL Core Operation APIs, 441
 OS_ApplicationShutdown
 OSAL Core Operation APIs, 441
 OS_ArgCallback_t
 common_types.h, 1700
 OS_bin_sem_prop_t, 813
 creator, 813
 name, 813
 value, 813
 OS_BinSemCreate
 OSAL Binary Semaphore APIs, 420
 OS_BinSemDelete
 OSAL Binary Semaphore APIs, 421
 OS_BinSemFlush
 OSAL Binary Semaphore APIs, 421
 OS_BinSemGetIdByName
 OSAL Binary Semaphore APIs, 422
 OS_BinSemGetInfo
 OSAL Binary Semaphore APIs, 422
 OS_BinSemGive
 OSAL Binary Semaphore APIs, 423
 OS_BinSemTake
 OSAL Binary Semaphore APIs, 423
 OS_BinSemTimedWait
 OSAL Binary Semaphore APIs, 424
 OS_BSP_GetArgC
 OSAL BSP low level access APIs, 425
 OS_BSP_GetArgV
 OSAL BSP low level access APIs, 425
 OS_BSP_GetResourceTypeConfig
 OSAL BSP low level access APIs, 425
 OS_BSP_SetExitCode
 OSAL BSP low level access APIs, 425
 OS_BSP_SetResourceTypeConfig
 OSAL BSP low level access APIs, 425
 OS_BUFFER_MSG_DEPTH
 osconfig.h, 1283
 OS_BUFFER_SIZE
 osconfig.h, 1283
 OS_BUILD_BASELINE
 osapi-version.h, 1735
 OS_BUILD_CODENAME
 osapi-version.h, 1735
 OS_BUILD_DEV_CYCLE
 osapi-version.h, 1735
 OS_BUILD_NUMBER
 osapi-version.h, 1735
 OS_CFG_MAX_VERSION_STR_LEN
 osapi-version.h, 1735
 OS_CHECK
 osapi-constants.h, 1709
 OS_CHK_ONLY
 osapi-fs.h, 1718
 OS_chkfs
 OSAL File System Level APIs, 483
 OS_chmod
 OSAL Standard File APIs, 470
 OS_close
 OSAL Standard File APIs, 471
 OS_CloseAllFiles
 OSAL Standard File APIs, 471
 OS_CloseFileByName
 OSAL Standard File APIs, 472
 OS_condvar_prop_t, 814
 creator, 814
 name, 814
 OS_CondVarBroadcast
 OSAL Condition Variable APIs, 444
 OS_CondVarCreate
 OSAL Condition Variable APIs, 445
 OS_CondVarDelete
 OSAL Condition Variable APIs, 446
 OS_CondVarGetIdByName
 OSAL Condition Variable APIs, 446
 OS_CondVarGetInfo
 OSAL Condition Variable APIs, 447
 OS_CondVarLock
 OSAL Condition Variable APIs, 447
 OS_CondVarSignal
 OSAL Condition Variable APIs, 447
 OS_CondVarTimedWait
 OSAL Condition Variable APIs, 448
 OS_CondVarUnlock
 OSAL Condition Variable APIs, 448
 OS_CondVarWait
 OSAL Condition Variable APIs, 449
 OS_ConvertToArrayIndex
 OSAL Object ID Utility APIs, 495
 OS_count_sem_prop_t, 814
 creator, 814
 name, 814
 value, 815

OS_CountSemCreate
 OSAL Counting Semaphore APIs, [450](#)
OS_CountSemDelete
 OSAL Counting Semaphore APIs, [451](#)
OS_CountSemGetIdByName
 OSAL Counting Semaphore APIs, [451](#)
OS_CountSemGetInfo
 OSAL Counting Semaphore APIs, [452](#)
OS_CountSemGive
 OSAL Counting Semaphore APIs, [452](#)
OS_CountSemTake
 OSAL Counting Semaphore APIs, [453](#)
OS_CountSemTimedWait
 OSAL Counting Semaphore APIs, [453](#)
OS_cp
 OSAL Standard File APIs, [472](#)
OS_DeleteAllObjects
 OSAL Core Operation APIs, [442](#)
OS_DirectoryClose
 OSAL Directory APIs, [455](#)
OS_DirectoryOpen
 OSAL Directory APIs, [455](#)
OS_DirectoryRead
 OSAL Directory APIs, [456](#)
OS_DirectoryRewind
 OSAL Directory APIs, [456](#)
os_dirent_t, [815](#)
 FileName, [815](#)
OS_DIRENTRY_NAME
 osapi-dir.h, [1711](#)
OS_ERR_BAD_ADDRESS
 OSAL Return Code Defines, [461](#)
OS_ERR_FILE
 OSAL Return Code Defines, [461](#)
OS_ERR_INCORRECT_OBJ_STATE
 OSAL Return Code Defines, [461](#)
OS_ERR_INCORRECT_OBJ_TYPE
 OSAL Return Code Defines, [461](#)
OS_ERR_INVALID_ARGUMENT
 OSAL Return Code Defines, [461](#)
OS_ERR_INVALID_ID
 OSAL Return Code Defines, [461](#)
OS_ERR_INVALID_PRIORITY
 OSAL Return Code Defines, [462](#)
OS_ERR_INVALID_SIZE
 OSAL Return Code Defines, [462](#)
OS_ERR_NAME_NOT_FOUND
 OSAL Return Code Defines, [462](#)
os_err_name_t
 osapi-error.h, [1714](#)
OS_ERR_NAME_TAKEN
 OSAL Return Code Defines, [462](#)
OS_ERR_NAME_TOO_LONG
 OSAL Return Code Defines, [462](#)
OS_ERR_NO_FREE_IDS
 OSAL Return Code Defines, [462](#)
OS_ERR_NOT_IMPLEMENTED
 OSAL Return Code Defines, [462](#)
OS_ERR_OBJECT_IN_USE
 OSAL Return Code Defines, [462](#)
OS_ERR_OPERATION_NOT_SUPPORTED
 OSAL Return Code Defines, [462](#)
OS_ERR_OUTPUT_TOO_LARGE
 OSAL Return Code Defines, [462](#)
OS_ERR_SEM_NOT_FULL
 OSAL Return Code Defines, [463](#)
OS_ERR_STREAM_DISCONNECTED
 OSAL Return Code Defines, [463](#)
OS_ERROR
 OSAL Return Code Defines, [463](#)
OS_ERROR_ADDRESS_MISALIGNED
 OSAL Return Code Defines, [463](#)
OS_ERROR_NAME_LENGTH
 osapi-error.h, [1713](#)
OS_ERROR_TIMEOUT
 OSAL Return Code Defines, [463](#)
OS_EVENT_MAX
 osapi-common.h, [1708](#)
OS_EVENT_RESERVED
 osapi-common.h, [1707](#)
OS_EVENT_RESOURCE_ALLOCATED
 osapi-common.h, [1707](#)
OS_EVENT_RESOURCE_CREATED
 osapi-common.h, [1707](#)
OS_EVENT_RESOURCE_DELETED
 osapi-common.h, [1708](#)
OS_Event_t
 osapi-common.h, [1707](#)
OS_EVENT_TASK_STARTUP
 osapi-common.h, [1708](#)
OS_EventHandler_t
 osapi-common.h, [1707](#)
OS_FDGetInfo
 OSAL Standard File APIs, [473](#)
OS_FdSet, [815](#)
 object_ids, [816](#)
OS_FILE_FLAG_CREATE
 osapi-file.h, [1717](#)
OS_FILE_FLAG_NONE
 osapi-file.h, [1717](#)
OS_file_flag_t
 osapi-file.h, [1717](#)
OS_FILE_FLAG_TRUNCATE
 osapi-file.h, [1717](#)
OS_file_prop_t, [816](#)
 IsValid, [816](#)
 Path, [816](#)
 User, [816](#)

OS_FileOpenCheck
 OSAL Standard File APIs, 473

OS_FILESTAT_EXEC
 osapi-file.h, 1716

OS_FILESTAT_ISDIR
 osapi-file.h, 1716

OS_FILESTAT_MODE
 osapi-file.h, 1716

OS_FILESTAT_MODE_DIR
 osapi-file.h, 1717

OS_FILESTAT_MODE_EXEC
 osapi-file.h, 1717

OS_FILESTAT_MODE_READ
 osapi-file.h, 1717

OS_FILESTAT_MODE_WRITE
 osapi-file.h, 1717

OS_FILESTAT_READ
 osapi-file.h, 1716

OS_FILESTAT_SIZE
 osapi-file.h, 1716

OS_FILESTAT_TIME
 osapi-file.h, 1716

OS_FILESTAT_WRITE
 osapi-file.h, 1717

OS_FileSysAddFixedMap
 OSAL File System Level APIs, 484

OS_FileSysStatVolume
 OSAL File System Level APIs, 484

OS_ForEachObject
 OSAL Object ID Utility APIs, 496

OS_ForEachObjectOfType
 OSAL Object ID Utility APIs, 496

OS_FP_ENABLED
 osapi-task.h, 1731

OS_FS_DEV_NAME_LEN
 osconfig.h, 1283

OS_FS_ERR_DEVICE_NOT_FREE
 OSAL Return Code Defines, 463

OS_FS_ERR_DRIVE_NOT_CREATED
 OSAL Return Code Defines, 463

OS_FS_ERR_NAME_TOO_LONG
 OSAL Return Code Defines, 463

OS_FS_ERR_PATH_INVALID
 OSAL Return Code Defines, 463

OS_FS_ERR_PATH_TOO_LONG
 OSAL Return Code Defines, 463

OS_FS_GetPhysDriveName
 OSAL File System Level APIs, 485

OS_FS_PHYS_NAME_LEN
 osconfig.h, 1283

OS_FS_VOL_NAME_LEN
 osconfig.h, 1283

os_fsinfo_t, 816
 FreeFds, 817

FreeVolumes, 817

MaxFds, 817

MaxVolumes, 817

os_fstat_t, 817
 FileModeBits, 818

FileSize, 818

FileTime, 818

OS_GetBuildNumber
 osapi-version.h, 1736

OS_GetErrorName
 OSAL Error Info APIs, 466

OS_GetFsInfo
 OSAL File System Level APIs, 486

OS_GetLocalTime
 OSAL Real Time Clock APIs, 427

OS_GetResourceName
 OSAL Object ID Utility APIs, 496

OS_GetVersionCodeName
 osapi-version.h, 1737

OS_GetVersionNumber
 osapi-version.h, 1737

OS_GetVersionString
 osapi-version.h, 1737

OS_heap_prop_t, 818
 free_blocks, 818

free_bytes, 818

largest_free_block, 819

OS_HeapGetInfo
 OSAL Heap APIs, 491

OS_IdentifyObject
 OSAL Object ID Utility APIs, 497

OS_IdleLoop
 OSAL Core Operation APIs, 442

OS_initfs
 OSAL File System Level APIs, 486

OS_INVALID_INT_NUM
 OSAL Return Code Defines, 464

OS_INVALID_POINTER
 OSAL Return Code Defines, 464

OS_INVALID_SEM_VALUE
 OSAL Return Code Defines, 464

OS_LAST_OFFICIAL
 osapi-version.h, 1735

OS_Iseek
 OSAL Standard File APIs, 474

OS_MAJOR_VERSION
 osapi-version.h, 1735

OS_MAX_API_NAME
 osconfig.h, 1283

OS_MAX_BIN_SEMAPHORES
 osconfig.h, 1283

OS_MAX_CMD_LEN
 osconfig.h, 1283

OS_MAX_CONDVARSL

osconfig.h, 1284
OS_MAX_CONSOLES
 osconfig.h, 1284
OS_MAX_COUNT_SEMAPHORES
 osconfig.h, 1284
OS_MAX_FILE_NAME
 osconfig.h, 1284
OS_MAX_FILE_SYSTEMS
 osconfig.h, 1284
OS_MAX_LOCAL_PATH_LEN
 osapi-constants.h, 1709
OS_MAX_MODULES
 osconfig.h, 1284
OS_MAX_MUTEXES
 osconfig.h, 1284
OS_MAX_NUM_OPEN_DIRS
 osconfig.h, 1285
OS_MAX_NUM_OPEN_FILES
 osconfig.h, 1285
OS_MAX_PATH_LEN
 osconfig.h, 1285
OS_MAX_QUEUES
 osconfig.h, 1285
OS_MAX_SYM_LEN
 osconfig.h, 1285
OS_MAX_TASK_PRIORITY
 osapi-task.h, 1731
OS_MAX_TASKS
 osconfig.h, 1285
OS_MAX_TIMEBASES
 osconfig.h, 1285
OS_MAX_TIMERS
 osconfig.h, 1286
OS_MINOR_VERSION
 osapi-version.h, 1736
OS_MISSION_REV
 osapi-version.h, 1736
OS_mkdir
 OSAL Directory APIs, 457
OS_mkfs
 OSAL File System Level APIs, 487
OS_module_address_t, 819
 bss_address, 819
 bss_size, 819
 code_address, 819
 code_size, 819
 data_address, 819
 data_size, 819
 flags, 820
 valid, 820
OS_MODULE_FILE_EXTENSION
 osconfig.h, 1286
OS_MODULE_FLAG_GLOBAL_SYMBOLS
 osapi-module.h, 1723
OS_MODULE_FLAG_LOCAL_SYMBOLS
 osapi-module.h, 1723
OS_module_prop_t, 820
 addr, 820
 entry_point, 820
 filename, 820
 host_module_id, 820
 name, 820
OS_ModuleInfo
 OSAL Dynamic Loader and Symbol APIs, 500
OS_ModuleLoad
 OSAL Dynamic Loader and Symbol APIs, 500
OS_ModuleSymbolLookup
 OSAL Dynamic Loader and Symbol APIs, 501
OS_ModuleUnload
 OSAL Dynamic Loader and Symbol APIs, 502
OS_mount
 OSAL File System Level APIs, 487
OS_mut_sem_prop_t, 821
 creator, 821
 name, 821
OS_MutSemCreate
 OSAL Mutex APIs, 504
OS_MutSemDelete
 OSAL Mutex APIs, 504
OS_MutSemGetIdByName
 OSAL Mutex APIs, 505
OS_MutSemGetInfo
 OSAL Mutex APIs, 505
OS_MutSemGive
 OSAL Mutex APIs, 506
OS_MutSemTake
 OSAL Mutex APIs, 506
OS_mv
 OSAL Standard File APIs, 474
OS_NetworkGetHostName
 OSAL Network ID APIs, 508
OS_NetworkGetID
 OSAL Network ID APIs, 508
OS_OBJECT_CREATOR_ANY
 osapi-constants.h, 1709
OS_OBJECT_ID_UNDEFINED
 osapi-constants.h, 1709
OS_OBJECT_INDEX_MASK
 osapi-idmap.h, 1720
OS_OBJECT_TYPE_OS_BINSEM
 OSAL Object Type Defines, 492
OS_OBJECT_TYPE_OS_CONDVAR
 OSAL Object Type Defines, 492
OS_OBJECT_TYPE_OS_CONSOLE
 OSAL Object Type Defines, 492
OS_OBJECT_TYPE_OS_COUNTSEM
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_DIR

OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_FILESYS
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_MODULE
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_MUTEX
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_QUEUE
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_STREAM
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_TASK
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_TIMEBASE
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_OS_TIMECB
 OSAL Object Type Defines, 493
OS_OBJECT_TYPE_SHIFT
 osapi-idmap.h, 1720
OS_OBJECT_TYPE_UNDEFINED
 OSAL Object Type Defines, 494
OS_OBJECT_TYPE_USER
 OSAL Object Type Defines, 494
OS_ObjectIdDefined
 OSAL Object ID Utility APIs, 497
OS_ObjectIdEqual
 OSAL Object ID Utility APIs, 498
OS_ObjectIdFromInteger
 OSAL Object ID Utility APIs, 498
OS_ObjectIdToArrayList
 OSAL Object ID Utility APIs, 498
OS_ObjectIdToInteger
 OSAL Object ID Utility APIs, 499
OS_OpenCreate
 OSAL Standard File APIs, 475
OS_PEND
 osapi-constants.h, 1709
OS_PRINTF
 cfe_es.h, 1371
 common_types.h, 1699
OS_printf
 OSAL Printf APIs, 510
OS_PRINTF_CONSOLE_NAME
 osconfig.h, 1286
OS_printf_disable
 OSAL Printf APIs, 510
OS_printf_enable
 OSAL Printf APIs, 510
OS_QUEUE_EMPTY
 OSAL Return Code Defines, 464
OS_QUEUE_FULL
 OSAL Return Code Defines, 464
OS_QUEUE_ID_ERROR
 OSAL Return Code Defines, 464
OS_QUEUE_INVALID_SIZE
 OSAL Return Code Defines, 464
OS_QUEUE_MAX_DEPTH
 osconfig.h, 1286
OS_queue_prop_t, 821
 creator, 821
 name, 822
OS_QUEUE_TIMEOUT
 OSAL Return Code Defines, 464
OS_QueueCreate
 OSAL Message Queue APIs, 511
OS_QueueDelete
 OSAL Message Queue APIs, 512
OS_QueueGet
 OSAL Message Queue APIs, 512
OS_QueueGetByName
 OSAL Message Queue APIs, 513
OS_QueueGetInfo
 OSAL Message Queue APIs, 513
OS_QueuePut
 OSAL Message Queue APIs, 514
OS_read
 OSAL Standard File APIs, 476
OS_READ_ONLY
 OSAL File Access Option Defines, 468
OS_READ_WRITE
 OSAL File Access Option Defines, 468
OS_RegisterEventHandler
 OSAL Core Operation APIs, 442
OS_remove
 OSAL Standard File APIs, 476
OS_rename
 OSAL Standard File APIs, 477
OS_REPAIR
 osapi-filesystem.h, 1718
OS_REVISION
 osapi-version.h, 1736
OS_rmdir
 OSAL Directory APIs, 457
OS_rmfs
 OSAL File System Level APIs, 488
OS_SEEK_CUR
 OSAL Reference Point For Seek Offset Defines, 469
OS_SEEK_END
 OSAL Reference Point For Seek Offset Defines, 469
OS_SEEK_SET
 OSAL Reference Point For Seek Offset Defines, 469
OS_SelectFdAdd
 OSAL Select APIs, 515
OS_SelectFdClear
 OSAL Select APIs, 515
OS_SelectFdIsSet
 OSAL Select APIs, 516
OS_SelectFdZero

OSAL Select APIs, 516
OS_SelectMultiple
 OSAL Select APIs, 517
OS_SelectMultipleAbs
 OSAL Select APIs, 518
OS_SelectSingle
 OSAL Select APIs, 519
OS_SelectSingleAbs
 OSAL Select APIs, 519
OS_SEM_EMPTY
 OSAL Semaphore State Defines, 419
OS_SEM_FAILURE
 OSAL Return Code Defines, 464
OS_SEM_FULL
 OSAL Semaphore State Defines, 419
OS_SEM_TIMEOUT
 OSAL Return Code Defines, 464
OS_SetLocalTime
 OSAL Real Time Clock APIs, 427
OS_SHELL_CMD_INPUT_FILE_NAME
 osconfig.h, 1286
OS_ShellOutputToFile
 OSAL Shell APIs, 521
OS SOCKADDR_MAX_LEN
 osapi-sockets.h, 1729
 osconfig.h, 1286
OS_SockAddr_t, 822
 ActualLength, 822
 AddrData, 822
OS_SockAddrData_t, 822
 AlignPtr, 823
 AlignU32, 823
 Buffer, 823
OS_socket_prop_t, 823
 creator, 824
 name, 824
OS_SocketAccept
 OSAL Socket Management APIs, 526
OS_SocketAcceptAbs
 OSAL Socket Management APIs, 527
OS_SocketAddrFromString
 OSAL Socket Address APIs, 522
OS_SocketAddrGetPort
 OSAL Socket Address APIs, 523
OS_SocketAddrInit
 OSAL Socket Address APIs, 523
OS_SocketAddrSetPort
 OSAL Socket Address APIs, 524
OS_SocketAddrToString
 OSAL Socket Address APIs, 524
OS_SocketBind
 OSAL Socket Management APIs, 528
OS_SocketBindAddress
 OSAL Socket Management APIs, 529
OS_SocketConnect
 OSAL Socket Management APIs, 529
OS_SocketConnectAbs
 OSAL Socket Management APIs, 530
OS_SocketDomain_INET
 osapi-sockets.h, 1729
OS_SocketDomain_INET6
 osapi-sockets.h, 1729
OS_SocketDomain_INVALID
 osapi-sockets.h, 1729
OS_SocketDomain_MAX
 osapi-sockets.h, 1729
OS_SocketDomain_t
 osapi-sockets.h, 1729
OS_SocketGetIdByName
 OSAL Socket Management APIs, 531
OS_SocketGetInfo
 OSAL Socket Management APIs, 531
OS_SocketListen
 OSAL Socket Management APIs, 532
OS_SocketOpen
 OSAL Socket Management APIs, 532
OS_SocketRecvFrom
 OSAL Socket Management APIs, 533
OS_SocketRecvFromAbs
 OSAL Socket Management APIs, 534
OS_SocketSendTo
 OSAL Socket Management APIs, 534
OS_SocketShutdown
 OSAL Socket Management APIs, 535
OS_SocketShutdownMode_NONE
 osapi-sockets.h, 1729
OS_SocketShutdownMode_SHUT_READ
 osapi-sockets.h, 1729
OS_SocketShutdownMode_SHUT_READWRITE
 osapi-sockets.h, 1729
OS_SocketShutdownMode_SHUT_WRITE
 osapi-sockets.h, 1729
OS_SocketShutdownMode_t
 osapi-sockets.h, 1729
OS_SocketType_DATAGRAM
 osapi-sockets.h, 1730
OS_SocketType_INVALID
 osapi-sockets.h, 1730
OS_SocketType_MAX
 osapi-sockets.h, 1730
OS_SocketType_STREAM
 osapi-sockets.h, 1730
OS_SocketType_t
 osapi-sockets.h, 1730
OS_stat
 OSAL Standard File APIs, 477
OS_static_symbol_record_t, 824
 Address, 824

Module, 824
Name, 824
OS_STATUS_STRING_LENGTH
 osapi-error.h, 1714
os_status_string_t
 osapi-error.h, 1714
OS_StatusToInteger
 OSAL Error Info APIs, 466
OS_StatusToString
 OSAL Error Info APIs, 467
OS_statvfs_t, 825
 block_size, 825
 blocks_free, 825
 total_blocks, 825
OS_STR
 osapi-version.h, 1736
OS_STR_HELPER
 osapi-version.h, 1736
OS_STREAM_STATE_BOUND
 osapi-select.h, 1727
OS_STREAM_STATE_CONNECTED
 osapi-select.h, 1727
OS_STREAM_STATE_LISTENING
 osapi-select.h, 1727
OS_STREAM_STATE_READABLE
 osapi-select.h, 1727
OS_STREAM_STATE_WRITABLE
 osapi-select.h, 1727
OS_StreamState_t
 osapi-select.h, 1727
OS_strlen
 OSAL Core Operation APIs, 442
OS_SUCCESS
 OSAL Return Code Defines, 465
OS_SymbolLookup
 OSAL Dynamic Loader and Symbol APIs, 502
OS_SymbolTableDump
 OSAL Dynamic Loader and Symbol APIs, 503
OS_task_prop_t, 825
 creator, 825
 name, 826
 priority, 826
 stack_size, 826
OS_TaskCreate
 OSAL Task APIs, 536
OS_TaskDelay
 OSAL Task APIs, 537
OS_TaskDelete
 OSAL Task APIs, 537
OS_TaskExit
 OSAL Task APIs, 538
OS_TaskFindIdBySystemData
 OSAL Task APIs, 538
OS_TaskGetId
 OSAL Task APIs, 539
OS_TaskGetIdByName
 OSAL Task APIs, 539
OS_TaskGetInfo
 OSAL Task APIs, 539
OS_TaskInstallDeleteHandler
 OSAL Task APIs, 540
OS_TaskSetPriority
 OSAL Task APIs, 540
OS_TIME_MAX
 osapi-clock.h, 1705
OS_TIME_MIN
 osapi-clock.h, 1705
OS_time_t, 826
 ticks, 826
OS_TIME_TICK_RESOLUTION_NS
 osapi-clock.h, 1706
OS_TIME_TICKS_PER_MSEC
 osapi-clock.h, 1706
OS_TIME_TICKS_PER_SECOND
 osapi-clock.h, 1706
OS_TIME_TICKS_PER_USEC
 osapi-clock.h, 1706
OS_TIME_ZERO
 osapi-clock.h, 1705
OS_TimeAdd
 OSAL Real Time Clock APIs, 428
OS_TimeAssembleFromMicroseconds
 OSAL Real Time Clock APIs, 428
OS_TimeAssembleFromMilliseconds
 OSAL Real Time Clock APIs, 428
OS_TimeAssembleFromNanoseconds
 OSAL Real Time Clock APIs, 429
OS_TimeAssembleFromSubseconds
 OSAL Real Time Clock APIs, 429
OS_timebase_prop_t, 827
 accuracy, 827
 creator, 827
 freerun_time, 827
 name, 827
 nominal_interval_time, 827
OS_TimeBaseCreate
 OSAL Time Base APIs, 542
OS_TimeBaseDelete
 OSAL Time Base APIs, 543
OS_TimeBaseGetFreeRun
 OSAL Time Base APIs, 543
OS_TimeBaseGetIdByName
 OSAL Time Base APIs, 544
OS_TimeBaseGetInfo
 OSAL Time Base APIs, 545
OS_TimeBaseSet
 OSAL Time Base APIs, 545
OS_TimeCompare

OSAL Real Time Clock APIs, 430
OS_TimedRead
 OSAL Standard File APIs, 478
OS_TimedReadAbs
 OSAL Standard File APIs, 479
OS_TimedWrite
 OSAL Standard File APIs, 480
OS_TimedWriteAbs
 OSAL Standard File APIs, 480
OS_TimeEqual
 OSAL Real Time Clock APIs, 431
OS_TimeFromRelativeMilliseconds
 OSAL Real Time Clock APIs, 431
OS_TimeFromTotalMicroseconds
 OSAL Real Time Clock APIs, 432
OS_TimeFromTotalMilliseconds
 OSAL Real Time Clock APIs, 432
OS_TimeFromTotalNanoseconds
 OSAL Real Time Clock APIs, 432
OS_TimeFromTotalSeconds
 OSAL Real Time Clock APIs, 433
OS_TimeGetFractionalPart
 OSAL Real Time Clock APIs, 433
OS_TimeGetMicrosecondsPart
 OSAL Real Time Clock APIs, 434
OS_TimeGetMillisecondsPart
 OSAL Real Time Clock APIs, 434
OS_TimeGetNanosecondsPart
 OSAL Real Time Clock APIs, 435
OS_TimeGetSign
 OSAL Real Time Clock APIs, 435
OS_TimeGetSubsecondsPart
 OSAL Real Time Clock APIs, 436
OS_TimeGetTotalMicroseconds
 OSAL Real Time Clock APIs, 436
OS_TimeGetTotalMilliseconds
 OSAL Real Time Clock APIs, 437
OS_TimeGetTotalNanoseconds
 OSAL Real Time Clock APIs, 437
OS_TimeGetTotalSeconds
 OSAL Real Time Clock APIs, 438
OS_TIMER_ERR_INTERNAL
 OSAL Return Code Defines, 465
OS_TIMER_ERR_INVALID_ARGS
 OSAL Return Code Defines, 465
OS_TIMER_ERR_TIMER_ID
 OSAL Return Code Defines, 465
OS_timer_prop_t, 827
 accuracy, 828
 creator, 828
 interval_time, 828
 name, 828
 start_time, 828
OS_TimerAdd
 OSAL Timer APIs, 547
OS_TimerCallback_t
 osapi-timer.h, 1734
OS_TimerCreate
 OSAL Timer APIs, 548
OS_TimerDelete
 OSAL Timer APIs, 549
OS_TimerGetIdByName
 OSAL Timer APIs, 550
OS_TimerGetInfo
 OSAL Timer APIs, 550
OS_TimerSet
 OSAL Timer APIs, 551
OS_TimerSync_t
 osapi-timebase.h, 1733
OS_TimeSubtract
 OSAL Real Time Clock APIs, 438
OS_TimeToRelativeMilliseconds
 OSAL Real Time Clock APIs, 438
OS_TranslatePath
 OSAL File System Level APIs, 488
OS_umount
 OSAL File System Level APIs, 489
OS_USED
 common_types.h, 1699
OS_UTILITYTASK_PRIORITY
 osconfig.h, 1286
OS_UTILITYTASK_STACK_SIZE
 osconfig.h, 1287
OS_VERSION
 osapi-version.h, 1736
OS_write
 OSAL Standard File APIs, 481
OS_WRITE_ONLY
 OSAL File Access Option Defines, 468
OSAL Binary Semaphore APIs, 420
 OS_BinSemCreate, 420
 OS_BinSemDelete, 421
 OS_BinSemFlush, 421
 OS_BinSemGetIdByName, 422
 OS_BinSemGetInfo, 422
 OS_BinSemGive, 423
 OS_BinSemTake, 423
 OS_BinSemTimedWait, 424
OSAL BSP low level access APIs, 425
 OS_BSP_GetArgC, 425
 OS_BSP_GetArgV, 425
 OS_BSP_GetResourceTypeConfig, 425
 OS_BSP_SetExitCode, 425
 OS_BSP_SetResourceTypeConfig, 425
OSAL Condition Variable APIs, 444
 OS_CondVarBroadcast, 444

OS_CondVarCreate, 445
 OS_CondVarDelete, 446
 OS_CondVarGetIdByName, 446
 OS_CondVarGetInfo, 447
 OS_CondVarLock, 447
 OS_CondVarSignal, 447
 OS_CondVarTimedWait, 448
 OS_CondVarUnlock, 448
 OS_CondVarWait, 449
OSAL Core Operation APIs, 440
 OS_API_Init, 440
 OS_API_Teardown, 441
 OS_Application_Run, 441
 OS_Application_Startup, 441
 OS_ApplicationExit, 441
 OS_ApplicationShutdown, 441
 OS_DeleteAllObjects, 442
 OS_IdleLoop, 442
 OS_RegisterEventHandler, 442
 OS_strlen, 442
OSAL Counting Semaphore APIs, 450
 OS_CountSemCreate, 450
 OS_CountSemDelete, 451
 OS_CountSemGetIdByName, 451
 OS_CountSemGetInfo, 452
 OS_CountSemGive, 452
 OS_CountSemTake, 453
 OS_CountSemTimedWait, 453
OSAL Directory APIs, 455
 OS_DirectoryClose, 455
 OS_DirectoryOpen, 455
 OS_DirectoryRead, 456
 OS_DirectoryRewind, 456
 OS_mkdir, 457
 OS_rmdir, 457
OSAL Dynamic Loader and Symbol APIs, 500
 OS_ModuleInfo, 500
 OS_ModuleLoad, 500
 OS_ModuleSymbolLookup, 501
 OS_ModuleUnload, 502
 OS_SymbolLookup, 502
 OS_SymbolTableDump, 503
OSAL Error Info APIs, 466
 OS_GetErrorName, 466
 OS_StatusToInteger, 466
 OS_StatusToString, 467
OSAL File Access Option Defines, 468
 OS_READ_ONLY, 468
 OS_READ_WRITE, 468
 OS_WRITE_ONLY, 468
OSAL File System Level APIs, 483
 OS_chkfs, 483
 OS_FileSysAddFixedMap, 484
 OS_FileSysStatVolume, 484
 OS_FS_GetPhysDriveName, 485
 OS_GetFsInfo, 486
 OS_initfs, 486
 OS_mkfs, 487
 OS_mount, 487
 OS_rmfs, 488
 OS_TranslatePath, 488
 OS_unmount, 489
OSAL Heap APIs, 491
 OS_HeapGetInfo, 491
OSAL Message Queue APIs, 511
 OS_QueueCreate, 511
 OS_QueueDelete, 512
 OS_QueueGet, 512
 OS_QueueGetIdByName, 513
 OS_QueueGetInfo, 513
 OS_QueuePut, 514
OSAL Mutex APIs, 504
 OS_MutSemCreate, 504
 OS_MutSemDelete, 504
 OS_MutSemGetIdByName, 505
 OS_MutSemGetInfo, 505
 OS_MutSemGive, 506
 OS_MutSemTake, 506
OSAL Network ID APIs, 508
 OS_NetworkGetHostName, 508
 OS_NetworkGetID, 508
OSAL Object ID Utility APIs, 495
 OS_ConvertToArrayIndex, 495
 OS_ForEachObject, 496
 OS_ForEachObjectType, 496
 OS_GetResourceName, 496
 OS_IdentifyObject, 497
 OS_ObjectIdDefined, 497
 OS_ObjectIdEqual, 498
 OS_ObjectIdFromInteger, 498
 OS_ObjectIdToArrayIndex, 498
 OS_ObjectIdToInteger, 499
OSAL Object Type Defines, 492
 OS_OBJECT_TYPE_OS_BINSEM, 492
 OS_OBJECT_TYPE_OS_CONDVAR, 492
 OS_OBJECT_TYPE_OS_CONSOLE, 492
 OS_OBJECT_TYPE_OS_COUNTSEM, 493
 OS_OBJECT_TYPE_OS_DIR, 493
 OS_OBJECT_TYPE_OS_FILESYS, 493
 OS_OBJECT_TYPE_OS_MODULE, 493
 OS_OBJECT_TYPE_OS_MUTEX, 493
 OS_OBJECT_TYPE_OS_QUEUE, 493
 OS_OBJECT_TYPE_OS_STREAM, 493
 OS_OBJECT_TYPE_OS_TASK, 493
 OS_OBJECT_TYPE_OS_TIMEBASE, 493
 OS_OBJECT_TYPE_OS_TIMECB, 493
 OS_OBJECT_TYPE_UNDEFINED, 494
 OS_OBJECT_TYPE_USER, 494

- OSAL Printf APIs, 510
 OS_printf, 510
 OS_printf_disable, 510
 OS_printf_enable, 510
- OSAL Real Time Clock APIs, 426
 OS_GetLocalTime, 427
 OS_SetLocalTime, 427
 OS_TimeAdd, 428
 OS_TimeAssembleFromMicroseconds, 428
 OS_TimeAssembleFromMilliseconds, 428
 OS_TimeAssembleFromNanoseconds, 429
 OS_TimeAssembleFromSubseconds, 429
 OS_TimeCompare, 430
 OS_TimeEqual, 431
 OS_TimeFromRelativeMilliseconds, 431
 OS_TimeFromTotalMicroseconds, 432
 OS_TimeFromTotalMilliseconds, 432
 OS_TimeFromTotalNanoseconds, 432
 OS_TimeFromTotalSeconds, 433
 OS_TimeGetFractionalPart, 433
 OS_TimeGetMicrosecondsPart, 434
 OS_TimeGetMillisecondsPart, 434
 OS_TimeGetNanosecondsPart, 435
 OS_TimeGetSign, 435
 OS_TimeGetSubsecondsPart, 436
 OS_TimeGetTotalMicroseconds, 436
 OS_TimeGetTotalMilliseconds, 437
 OS_TimeGetTotalNanoseconds, 437
 OS_TimeGetTotalSeconds, 438
 OS_TimeSubtract, 438
 OS_TimeToRelativeMilliseconds, 438
- OSAL Reference Point For Seek Offset Defines, 469
 OS_SEEK_CUR, 469
 OS_SEEK_END, 469
 OS_SEEK_SET, 469
- OSAL Return Code Defines, 459
 OS_ERR_BAD_ADDRESS, 461
 OS_ERR_FILE, 461
 OS_ERR_INCORRECT_OBJ_STATE, 461
 OS_ERR_INCORRECT_OBJ_TYPE, 461
 OS_ERR_INVALID_ARGUMENT, 461
 OS_ERR_INVALID_ID, 461
 OS_ERR_INVALID_PRIORITY, 462
 OS_ERR_INVALID_SIZE, 462
 OS_ERR_NAME_NOT_FOUND, 462
 OS_ERR_NAME_TAKEN, 462
 OS_ERR_NAME_TOO_LONG, 462
 OS_ERR_NO_FREE_IDS, 462
 OS_ERR_NOT_IMPLEMENTED, 462
 OS_ERR_OBJECT_IN_USE, 462
 OS_ERR_OPERATION_NOT_SUPPORTED, 462
 OS_ERR_OUTPUT_TOO_LARGE, 462
 OS_ERR_SEM_NOT_FULL, 463
 OS_ERR_STREAM_DISCONNECTED, 463
- OS_ERROR, 463
OS_ERROR_ADDRESS_MISALIGNED, 463
OS_ERROR_TIMEOUT, 463
OS_FS_ERR_DEVICE_NOT_FREE, 463
OS_FS_ERR_DRIVE_NOT_CREATED, 463
OS_FS_ERR_NAME_TOO_LONG, 463
OS_FS_ERR_PATH_INVALID, 463
OS_FS_ERR_PATH_TOO_LONG, 463
OS_INVALID_INT_NUM, 464
OS_INVALID_POINTER, 464
OS_INVALID_SEM_VALUE, 464
OS_QUEUE_EMPTY, 464
OS_QUEUE_FULL, 464
OS_QUEUE_ID_ERROR, 464
OS_QUEUE_INVALID_SIZE, 464
OS_QUEUE_TIMEOUT, 464
OS_SEM_FAILURE, 464
OS_SEM_TIMEOUT, 464
OS_SUCCESS, 465
OS_TIMER_ERR_INTERNAL, 465
OS_TIMER_ERR_INVALID_ARGS, 465
OS_TIMER_ERR_TIMER_ID, 465
OS_TIMER_ERR_UNAVAILABLE, 465
- OSAL Select APIs, 515
 OS_SelectFdAdd, 515
 OS_SelectFdClear, 515
 OS_SelectFdsSet, 516
 OS_SelectFdZero, 516
 OS_SelectMultiple, 517
 OS_SelectMultipleAbs, 518
 OS_SelectSingle, 519
 OS_SelectSingleAbs, 519
- OSAL Semaphore State Defines, 419
 OS_SEM_EMPTY, 419
 OS_SEM_FULL, 419
- OSAL Shell APIs, 521
 OS_ShellOutputToFile, 521
- OSAL Socket Address APIs, 522
 OS_SocketAddrFromString, 522
 OS_SocketAddrGetPort, 523
 OS_SocketAddrInit, 523
 OS_SocketAddrSetPort, 524
 OS_SocketAddrToString, 524
- OSAL Socket Management APIs, 526
 OS_SocketAccept, 526
 OS_SocketAcceptAbs, 527
 OS_SocketBind, 528
 OS_SocketBindAddress, 529
 OS_SocketConnect, 529
 OS_SocketConnectAbs, 530
 OS_SocketGetIdByName, 531
 OS_SocketGetInfo, 531
 OS_SocketListen, 532
 OS_SocketOpen, 532

OS_SocketRecvFrom, 533
OS_SocketRecvFromAbs, 534
OS_SocketSendTo, 534
OS_SocketShutdown, 535
OSAL Standard File APIs, 470
 OS_chmod, 470
 OS_close, 471
 OS_CloseAllFiles, 471
 OS_CloseFileByName, 472
 OS_cp, 472
 OS_FDGetInfo, 473
 OS_FileOpenCheck, 473
 OS_lseek, 474
 OS_mv, 474
 OS_OpenCreate, 475
 OS_read, 476
 OS_remove, 476
 OS_rename, 477
 OS_stat, 477
 OS_TimedRead, 478
 OS_TimedReadAbs, 479
 OS_TimedWrite, 480
 OS_TimedWriteAbs, 480
 OS_write, 481
OSAL Task APIs, 536
 OS_TaskCreate, 536
 OS_TaskDelay, 537
 OS_TaskDelete, 537
 OS_TaskExit, 538
 OS_TaskFindIdBySystemData, 538
 OS_TaskGetId, 539
 OS_TaskGetIdByName, 539
 OS_TaskGetInfo, 539
 OS_TaskInstallDeleteHandler, 540
 OS_TaskSetPriority, 540
OSAL Time Base APIs, 542
 OS_TimeBaseCreate, 542
 OS_TimeBaseDelete, 543
 OS_TimeBaseGetFreeRun, 543
 OS_TimeBaseGetIdByName, 544
 OS_TimeBaseGetInfo, 545
 OS_TimeBaseSet, 545
OSAL Timer APIs, 547
 OS_TimerAdd, 547
 OS_TimerCreate, 548
 OS_TimerDelete, 549
 OS_TimerGetIdByName, 550
 OS_TimerGetInfo, 550
 OS_TimerSet, 551
osal/docs/src/osal_frontpage.dox, 1697
osal/docs/src/osal_fs.dox, 1697
osal/docs/src/osal_timer.dox, 1697
osal/src/os/inc/common_types.h, 1697
osal/src/os/inc/osapi-binsem.h, 1702
osal/src/os/inc/osapi-bsp.h, 1703
osal/src/os/inc/osapi-clock.h, 1703
osal/src/os/inc/osapi-common.h, 1706
osal/src/os/inc/osapi-condvar.h, 1708
osal/src/os/inc/osapi-constants.h, 1709
osal/src/os/inc/osapi-countsem.h, 1710
osal/src/os/inc/osapi-dir.h, 1710
osal/src/os/inc/osapi-error.h, 1711
osal/src/os/inc/osapi-file.h, 1714
osal/src/os/inc/osapi-fs.h, 1717
osal/src/os/inc/osapi-heap.h, 1719
osal/src/os/inc/osapi-idmap.h, 1719
osal/src/os/inc/osapi-macros.h, 1721
osal/src/os/inc/osapi-module.h, 1722
osal/src/os/inc/osapi-mutex.h, 1724
osal/src/os/inc/osapi-network.h, 1724
osal/src/os/inc/osapi-printf.h, 1725
osal/src/os/inc/osapi-queue.h, 1725
osal/src/os/inc/osapi-select.h, 1726
osal/src/os/inc/osapi-shell.h, 1727
osal/src/os/inc/osapi-sockets.h, 1727
osal/src/os/inc/osapi-task.h, 1730
osal/src/os/inc/osapi-timebase.h, 1732
osal/src/os/inc/osapi-timer.h, 1733
osal/src/os/inc/osapi-version.h, 1734
osal/src/os/inc/osapi.h, 1737
OSAL_API_VERSION
 osapi-version.h, 1736
OSAL_BLOCKCOUNT_C
 common_types.h, 1699
osal_blockcount_t
 common_types.h, 1700
OSAL_CONFIG_CONSOLE_ASYNC
 osconfig.h, 1287
OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER
 osconfig.h, 1287
OSAL_CONFIG_INCLUDE_NETWORK
 osconfig.h, 1287
OSAL_CONFIG_INCLUDE_STATIC_LOADER
 osconfig.h, 1287
osal_id_t
 common_types.h, 1700
OSAL_INDEX_C
 common_types.h, 1699
osal_index_t
 common_types.h, 1700
OSAL_OBJTYPE_C
 common_types.h, 1699
osal_objtype_t
 common_types.h, 1701
OSAL_PRIORITY_C
 osapi-task.h, 1731
osal_priority_t
 osapi-task.h, 1732

OSAL_SIZE_C
common_types.h, 1699

OSAL_STACKPTR_C
osapi-task.h, 1731

osal_stackptr_t
osapi-task.h, 1732

OSAL_STATUS_C
common_types.h, 1699

osal_status_t
common_types.h, 1701

osal_task
osapi-task.h, 1732

OSAL_TASK_STACK_ALLOCATE
osapi-task.h, 1731

OSALMajorVersion
CFE_ES_HousekeepingTlm_Payload, 677

OSALMinorVersion
CFE_ES_HousekeepingTlm_Payload, 677

OSALMissionRevision
CFE_ES_HousekeepingTlm_Payload, 677

OSALRevision
CFE_ES_HousekeepingTlm_Payload, 677

osapi-clock.h
OS_TIME_MAX, 1705
OS_TIME_MIN, 1705
OS_TIME_TICK_RESOLUTION_NS, 1706
OS_TIME_TICKS_PER_MSEC, 1706
OS_TIME_TICKS_PER_SECOND, 1706
OS_TIME_TICKS_PER_USEC, 1706
OS_TIME_ZERO, 1705

osapi-common.h
OS_EVENT_MAX, 1708
OS_EVENT_RESERVED, 1707
OS_EVENT_RESOURCE_ALLOCATED, 1707
OS_EVENT_RESOURCE_CREATED, 1707
OS_EVENT_RESOURCE_DELETED, 1708
OS_Event_t, 1707
OS_EVENT_TASK_STARTUP, 1708
OS_EventHandler_t, 1707

osapi-constants.h
OS_CHECK, 1709
OS_MAX_LOCAL_PATH_LEN, 1709
OS_OBJECT_CREATOR_ANY, 1709
OS_OBJECT_ID_UNDEFINED, 1709
OS_PEND, 1709

osapi-dir.h
OS_DIRENTRY_NAME, 1711

osapi-error.h
os_err_name_t, 1714
OS_ERROR_NAME_LENGTH, 1713
OS_STATUS_STRING_LENGTH, 1714
os_status_string_t, 1714

osapi-file.h
OS_FILE_FLAG_CREATE, 1717

OS_FILE_FLAG_NONE, 1717
OS_file_flag_t, 1717

OS_FILE_FLAG_TRUNCATE, 1717

OS_FILESTAT_EXEC, 1716

OS_FILESTAT_ISDIR, 1716

OS_FILESTAT_MODE, 1716

OS_FILESTAT_MODE_DIR, 1717

OS_FILESTAT_MODE_EXEC, 1717

OS_FILESTAT_MODE_READ, 1717

OS_FILESTAT_MODE_WRITE, 1717

OS_FILESTAT_READ, 1716

OS_FILESTAT_SIZE, 1716

OS_FILESTAT_TIME, 1716

OS_FILESTAT_WRITE, 1717

osapi-filesys.h
OS_CHK_ONLY, 1718
OS_REPAIR, 1718

osapi-idmap.h
OS_OBJECT_INDEX_MASK, 1720
OS_OBJECT_TYPE_SHIFT, 1720

osapi-macros.h
ARGCHECK, 1721
BUGCHECK, 1721
BUGCHECK_VOID, 1722
BUGREPORT, 1722
LENGTHCHECK, 1722

osapi-module.h
OS_MODULE_FLAG_GLOBAL_SYMBOLS, 1723
OS_MODULE_FLAG_LOCAL_SYMBOLS, 1723

osapi-select.h
OS_STREAM_STATE_BOUND, 1727
OS_STREAM_STATE_CONNECTED, 1727
OS_STREAM_STATE_LISTENING, 1727
OS_STREAM_STATE_READABLE, 1727
OS_STREAM_STATE_WRITABLE, 1727
OS_StreamState_t, 1727

osapi-sockets.h
OS SOCKADDR_MAX_LEN, 1729
OS_SocketDomain_INET, 1729
OS_SocketDomain_INET6, 1729
OS_SocketDomain_INVALID, 1729
OS_SocketDomain_MAX, 1729
OS_SocketDomain_t, 1729
OS_SocketShutdownMode_NONE, 1729
OS_SocketShutdownMode_SHUT_READ, 1729
OS_SocketShutdownMode_SHUT_READWRITE, 1729
OS_SocketShutdownMode_SHUT_WRITE, 1729
OS_SocketShutdownMode_t, 1729
OS_SocketType_DATAGRAM, 1730
OS_SocketType_INVALID, 1730
OS_SocketType_MAX, 1730
OS_SocketType_STREAM, 1730
OS_SocketType_t, 1730

osapi-task.h
 OS_FP_ENABLED, 1731
 OS_MAX_TASK_PRIORITY, 1731
 OSAL_PRIORITY_C, 1731
 osal_priority_t, 1732
 OSAL_STACKPTR_C, 1731
 osal_stackptr_t, 1732
 osal_task, 1732
 OSAL_TASK_STACK_ALLOCATE, 1731
 osapi-timebase.h
 OS_TimerSync_t, 1733
 osapi-timer.h
 OS_TimerCallback_t, 1734
 osapi-version.h
 OS_BUILD_BASELINE, 1735
 OS_BUILD_CODENAME, 1735
 OS_BUILD_DEV_CYCLE, 1735
 OS_BUILD_NUMBER, 1735
 OS_CFG_MAX_VERSION_STR_LEN, 1735
 OS_GetBuildNumber, 1736
 OS_GetVersionCodeName, 1737
 OS_GetVersionNumber, 1737
 OS_GetVersionString, 1737
 OS_LAST_OFFICIAL, 1735
 OS_MAJOR_VERSION, 1735
 OS_MINOR_VERSION, 1736
 OS_MISSION_REV, 1736
 OS_REVISION, 1736
 OS_STR, 1736
 OS_STR_HELPER, 1736
 OS_VERSION, 1736
 OSAL_API_VERSION, 1736
 osconfig.h
 OS_ADD_TASK_FLAGS, 1283
 OS_BUFFER_MSG_DEPTH, 1283
 OS_BUFFER_SIZE, 1283
 OS_FS_DEV_NAME_LEN, 1283
 OS_FS_PHYS_NAME_LEN, 1283
 OS_FS_VOL_NAME_LEN, 1283
 OS_MAX_API_NAME, 1283
 OS_MAX_BIN_SEMAPHORES, 1283
 OS_MAX_CMD_LEN, 1283
 OS_MAX_CONDVARs, 1284
 OS_MAX_CONSOLES, 1284
 OS_MAX_COUNT_SEMAPHORES, 1284
 OS_MAX_FILE_NAME, 1284
 OS_MAX_FILE_SYSTEMS, 1284
 OS_MAX_MODULES, 1284
 OS_MAX_MUTEXES, 1284
 OS_MAX_NUM_OPEN_DIRS, 1285
 OS_MAX_NUM_OPEN_FILES, 1285
 OS_MAX_PATH_LEN, 1285
 OS_MAX_QUEUES, 1285
 OS_MAX_SYM_LEN, 1285
 OS_MAX_TASKS, 1285
 OS_MAX_TIMEBASES, 1285
 OS_MAX_TIMERS, 1286
 OS_MODULE_FILE_EXTENSION, 1286
 OS_PRINTF_CONSOLE_NAME, 1286
 OS_QUEUE_MAX_DEPTH, 1286
 OS_SHELL_CMD_INPUT_FILE_NAME, 1286
 OS SOCKADDR_MAX_LEN, 1286
 OS_UTILITYTASK_PRIORITY, 1286
 OS_UTILITYTASK_STACK_SIZE, 1287
 OSAL_CONFIG_CONSOLE_ASYNC, 1287
 OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER, 1287
 OSAL_CONFIG_INCLUDE_NETWORK, 1287
 OSAL_CONFIG_INCLUDE_STATIC_LOADER, 1287
 out
 CF_Engine, 589
 outgoing_counter
 CF_Engine, 589
 outgoing_file_chunk_size
 CF_ConfigTable, 582
 OutputPort
 CFE_EVS_HousekeepingTlm_Payload, 719
 OwnerAppName
 CFE_TBL_TblRegPacket_Payload, 782
 PacketID
 CFE_EVS_LongEventTlm_Payload, 721
 CFE_EVS_ShortEventTlm_Payload, 730
 Parameter
 CFE_TBL_NotifyCmd_Payload, 777
 Path
 OS_file_prop_t, 816
 Payload
 CF_AbandonCmd, 554
 CF_CancelCmd, 556
 CF_DisableDequeueCmd, 584
 CF_DisableDirPollingCmd, 585
 CF_EnableDequeueCmd, 586
 CF_EnableDirPollingCmd, 587
 CF_EotPacket, 590
 CF_FreezeCmd, 596
 CF_GetParamCmd, 598
 CF_HkPacket, 606
 CF_PlaybackDirCmd, 631
 CF_PurgeQueueCmd, 634
 CF_ResetCountersCmd, 634
 CF_ResumeCmd, 635
 CF_SetParamCmd, 639
 CF_SuspendCmd, 641
 CF_ThawCmd, 642
 CF_TxFileCmd, 653
 CF_WriteQueueCmd, 658
 CFE_ES_DeleteCDSCmd, 669

CFE_ES_DumpCDSRegistryCmd, 670
CFE_ES_FileNameCmd, 671
CFE_ES_HousekeepingTlm, 672
CFE_ES_MemStatsTlm, 682
CFE_ES_OneAppTlm, 683
CFE_ES_OverWriteSysLogCmd, 684
CFE_ES_QueryAllCmd, 687
CFE_ES_QueryAllTasksCmd, 688
CFE_ES_QueryOneCmd, 688
CFE_ES_ReloadAppCmd, 689
CFE_ES_RestartAppCmd, 690
CFE_ES_RestartCmd, 691
CFE_ES_SendMemPoolStatsCmd, 692
CFE_ES_SetMaxPRCountCmd, 694
CFE_ES_SetPerfFilterMaskCmd, 695
CFE_ES_SetPerfTriggerMaskCmd, 696
CFE_ES_StartApp, 697
CFE_ES_StartPerfDataCmd, 700
CFE_ES_StopAppCmd, 700
CFE_ES_StopPerfDataCmd, 701
CFE_ES_WriteERLogCmd, 703
CFE_ES_WriteSysLogCmd, 704
CFE_EVS_AddEventFilterCmd, 704
CFE_EVS_DeleteEventFilterCmd, 711
CFE_EVS_DisableAppEventsCmd, 711
CFE_EVS_DisableAppEventTypeCmd, 712
CFE_EVS_DisableEventTypeCmd, 713
CFE_EVS_DisablePortsCmd, 713
CFE_EVS_EnableAppEventsCmd, 714
CFE_EVS_EnableAppEventTypeCmd, 714
CFE_EVS_EnableEventTypeCmd, 715
CFE_EVS_EnablePortsCmd, 716
CFE_EVS_HousekeepingTlm, 716
CFE_EVS_LongEventTlm, 720
CFE_EVS_ResetAllFiltersCmd, 724
CFE_EVS_ResetAppCounterCmd, 724
CFE_EVS_ResetFilterCmd, 725
CFE_EVS_SetEventFormatModeCmd, 727
CFE_EVS_SetFilterCmd, 728
CFE_EVS_SetLogModeCmd, 729
CFE_EVS_ShortEventTlm, 729
CFE_EVS_WriteAppDataFileCmd, 731
CFE_EVS_WriteLogFileCmd, 731
CFE_SB_AllSubscriptionsTlm, 734
CFE_SB_DisableRouteCmd, 736
CFE_SB_EnableRouteCmd, 737
CFE_SB_HousekeepingTlm, 738
CFE_SB_SingleSubscriptionTlm, 752
CFE_SB_StatsTlm, 753
CFE_SB_WriteMapInfoCmd, 758
CFE_SB_WritePipeInfoCmd, 759
CFE_SB_WriteRoutingInfoCmd, 760
CFE_TBL_AbortLoadCmd, 760
CFE_TBL_ActivateCmd, 761
CFE_TBL_DeleteCDSCmd, 763
CFE_TBL_DumpCmd, 764
CFE_TBL_DumpRegistryCmd, 765
CFE_TBL_HousekeepingTlm, 768
CFE_TBL_LoadCmd, 775
CFE_TBL_NotifyCmd, 776
CFE_TBL_SendRegistryCmd, 779
CFE_TBL_TableRegistryTlm, 780
CFE_TBL_ValidateCmd, 784
CFE_TIME_AddAdjustCmd, 785
CFE_TIME_AddDelayCmd, 785
CFE_TIME_AddOneHzAdjustmentCmd, 786
CFE_TIME_DiagnosticTlm, 787
CFE_TIME_HousekeepingTlm, 796
CFE_TIME_SetLeapSecondsCmd, 802
CFE_TIME_SetMETCmd, 803
CFE_TIME_SetSignalCmd, 804
CFE_TIME_SetSourceCmd, 804
CFE_TIME_SetStateCmd, 805
CFE_TIME_SetSTCFCmd, 805
CFE_TIME_SetTimeCmd, 806
CFE_TIME_SubAdjustCmd, 808
CFE_TIME_SubDelayCmd, 809
CFE_TIME_SubOneHzAdjustmentCmd, 809
CFE_TIME_ToneDataCmd, 811
pb
 CF_Poll, 632
 CF_Transaction, 646
pdec
 CF_Logical_PduBuffer, 614
pdu
 CF_HkRecv, 608
 CF_HkSent, 609
pdu_header
 CF_Logical_PduBuffer, 614
pdu_type
 CF_Logical_PduHeader, 620
PeakMemInUse
 CFE_SB_StatsTlm_Payload, 755
PeakMsgIdsInUse
 CFE_SB_StatsTlm_Payload, 755
PeakPipesInUse
 CFE_SB_StatsTlm_Payload, 756
PeakQueueDepth
 CFE_SB_PipeDepthStats, 745
 CFE_SB_PipeInfoEntry, 746
PeakSBBuffersInUse
 CFE_SB_StatsTlm_Payload, 756
PeakSubscriptionsInUse
 CFE_SB_StatsTlm_Payload, 756
peer_eid
 CF_EotPacket_Payload, 591
 CF_History, 599
penc

CF_Logical_PduBuffer, 614
 PerfDataCount
 CFE_ES_HousekeepingTlm_Payload, 677
 PerfDataEnd
 CFE_ES_HousekeepingTlm_Payload, 677
 PerfDataStart
 CFE_ES_HousekeepingTlm_Payload, 677
 PerfDataToWrite
 CFE_ES_HousekeepingTlm_Payload, 678
 PerfFilterMask
 CFE_ES_HousekeepingTlm_Payload, 678
 PerfMode
 CFE_ES_HousekeepingTlm_Payload, 678
 PerfState
 CFE_ES_HousekeepingTlm_Payload, 678
 PerfTriggerCount
 CFE_ES_HousekeepingTlm_Payload, 678
 PerfTriggerMask
 CFE_ES_HousekeepingTlm_Payload, 678
 ph
 CF_PduCmdMsg, 628
 CF_PduTlmMsg, 629
 Pipe
 CFE_SB_RouteCmd_Payload, 749
 CFE_SB_SingleSubscriptionTlm_Payload, 752
 CFE_SB_SubEntries, 757
 pipe
 CF_Channel, 573
 pipe_depth_input
 CF_ChannelConfig, 576
 PipeDepthStats
 CFE_SB_StatsTlm_Payload, 756
 Pipeld
 CFE_SB_PipeDepthStats, 745
 CFE_SB_PipeInfoEntry, 747
 CFE_SB_RoutingFileEntry, 750
 PipeName
 CFE_SB_PipeInfoEntry, 747
 CFE_SB_RoutingFileEntry, 750
 PipeOptsErrorCounter
 CFE_ES_HousekeepingTlm_Payload, 741
 PipeOverflowErrorCounter
 CFE_ES_HousekeepingTlm_Payload, 741
 PipesInUse
 CFE_SB_StatsTlm_Payload, 756
 PktSegment
 CFE_SB_AllSubscriptionsTlm_Payload, 735
 playback
 CF_Channel, 573
 playback_counter
 CF_HkChannel_Data, 601
 poll
 CF_Channel, 574
 poll_counter

CF_HkChannel_Data, 601
 polldir
 CF_ChannelConfig, 576
 PoolHandle
 CFE_ES_PoolStatsTlm_Payload, 686
 CFE_ES_SendMemPoolStatsCmd_Payload, 693
 PoolSize
 CFE_ES_MemPoolStats, 682
 PoolStats
 CFE_ES_PoolStatsTlm_Payload, 686
 prev
 CF_CListNode, 579
 Priority
 CFE_ES_AppInfo, 664
 CFE_ES_StartAppCmd_Payload, 698
 CFE_ES_TaskInfo, 702
 CFE_SB_Qos_t, 747
 priority
 CF_Playback, 630
 CF_PollDir, 633
 CF_Transaction, 646
 CF_Traverse_PriorityArg, 648
 CF_TxFile_Payload, 653
 OS_task_prop_t, 826
 ProcessorID
 CFE_EVS_PacketID, 723
 CFE_FS_Header, 734
 ProcessorResets
 CFE_ES_HousekeepingTlm_Payload, 678
 psp/fsw/inc/cfe_psp.h, 1738
 psp/fsw/inc/cfe_psp_cache_api.h, 1739
 psp/fsw/inc/cfe_psp_cds_api.h, 1739
 psp/fsw/inc/cfe_psp_eepromaccess_api.h, 1741
 psp/fsw/inc/cfe_psp_error.h, 1744
 psp/fsw/inc/cfe_psp_exception_api.h, 1747
 psp/fsw/inc/cfe_psp_id_api.h, 1749
 psp/fsw/inc/cfe_psp_memaccess_api.h, 1749
 psp/fsw/inc/cfe_psp_memrange_api.h, 1753
 psp/fsw/inc/cfe_psp_port_api.h, 1759
 psp/fsw/inc/cfe_psp_ssri_api.h, 1761
 psp/fsw/inc/cfe_psp_timertick_api.h, 1762
 psp/fsw/inc/cfe_psp_version_api.h, 1764
 psp/fsw/inc/cfe_psp_watchdog_api.h, 1765
 PSPMajorVersion
 CFE_ES_HousekeepingTlm_Payload, 679
 PSPMinorVersion
 CFE_ES_HousekeepingTlm_Payload, 679
 PSPMissionRevision
 CFE_ES_HousekeepingTlm_Payload, 679
 PSPRevision
 CFE_ES_HousekeepingTlm_Payload, 679
 Ptr
 CFE_ES_PoolAlign, 686

q_index
 CF_Flags_Common, 593

q_size
 CF_HkChannel_Data, 601

Qos
 CFE_SB_SingleSubscriptionTlm_Payload, 753

 CFE_SB_SubEntries, 757

qs
 CF_Channel, 574

queue
 CF_WriteQueue_Payload, 658

r2
 CF_RxState_Data, 637

ran_one
 CF_CFDP_CycleTx_args, 557

receive
 CF_StateData, 640

recv
 CF_HkCounters, 603

RegisteredCoreApps
 CFE_ES_HousekeepingTlm_Payload, 679

RegisteredExternalApps
 CFE_ES_HousekeepingTlm_Payload, 679

RegisteredLibs
 CFE_ES_HousekeepingTlm_Payload, 679

RegisteredTasks
 CFE_ES_HousekeepingTlm_Payload, 680

Reliability
 CFE_SB_Qos_t, 747

Reserved
 CFE_TBL_File_Hdr, 766

ResetSubtype
 CFE_ES_HousekeepingTlm_Payload, 680

ResetType
 CFE_ES_HousekeepingTlm_Payload, 680

ResourceId
 CFE_ES_AppInfo, 664

RestartType
 CFE_ES_RestartCmd_Payload, 691

result
 CF_Crc, 583

RunStatus
 CF_AppData_t, 556

rx
 CF_CFDP_TxnRecvDispatchTable_t, 568

 CF_StateFlags, 641

rx_crc_calc_bytes
 CF_RxS2_Data, 636

rx_crc_calc_bytes_per_wakeup
 CF_ConfigTable, 582

rx_max_messages_per_wakeup
 CF_ChannelConfig, 576

rx_pdudata

CF_Input, 610

s2

CF_TxState_Data, 655

same
 CF_ChanAction_SuspResArg, 572

sample_perfids.h
 CFE_MISSION_ES_MAIN_PERF_ID, 1343

 CFE_MISSION_ES_PERF_EXIT_BIT, 1343

 CFE_MISSION_EVS_MAIN_PERF_ID, 1343

 CFE_MISSION_SB_MAIN_PERF_ID, 1343

 CFE_MISSION_SB_MSG_LIM_PERF_ID, 1343

 CFE_MISSION_SB_PIPE_OFLOW_PERF_ID, 1343

 CFE_MISSION_TBL_MAIN_PERF_ID, 1343

 CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID, 1343

 CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID, 1343

 CFE_MISSION_TIME_MAIN_PERF_ID, 1344

 CFE_MISSION_TIME_SENDMET_PERF_ID, 1344

 CFE_MISSION_TIME_TONE1HZISR_PERF_ID, 1344

 CFE_MISSION_TIME_TONE1HZTASK_PERF_ID, 1344

SBBuffersInUse
 CFE_SB_StatsTlm_Payload, 756

scope_end
 CF_CFDP_PduNak, 564

 CF_Logical_PduNak, 622

scope_start
 CF_CFDP_PduNak, 564

 CF_Logical_PduNak, 622

Seconds
 CFE_TIME_OneHzAdjustmentCmd_Payload, 800

 CFE_TIME_SysTime, 810

 CFE_TIME_TimeCmd_Payload, 811

Seconds1HzAdj
 CFE_TIME_HousekeepingTlm_Payload, 798

SecondsDelay
 CFE_TIME_HousekeepingTlm_Payload, 798

SecondsMET
 CFE_TIME_HousekeepingTlm_Payload, 798

SecondsSTCF
 CFE_TIME_HousekeepingTlm_Payload, 798

segment_list
 CF_Logical_PduFileDataHeader, 616

 CF_Logical_PduNak, 622

segment_meta_flag
 CF_Logical_PduHeader, 620

segmentation_control
 CF_CFDP_PduMd, 563

segments
 CF_Logical_SegmentList, 623

sem_id

CF_Channel, 574
 sem_name
 CF_ChannelConfig, 576
 send
 CF_StateData, 640
 send_ack
 CF_Flags_Rx, 594
 send_fin
 CF_Flags_Rx, 594
 send_nak
 CF_Flags_Rx, 595
 SendErrors
 CFE_SB_PipeInfoEntry, 747
 sent
 CF_HkCounters, 603
 seq_num
 CF_Engine, 589
 CF_EotPacket_Payload, 592
 CF_History, 599
 Sequence
 CCSDS_PrimaryHeader, 554
 sequence_num
 CF_Logical_PduHeader, 620
 ServerFlyState
 CFE_TIME_DiagnosticTlm_Payload, 793
 shift
 CF_Codec_BitField, 580
 Size
 CFE_ES_CDSRegDumpRec, 668
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 782
 size
 CF_CFDP_PduEof, 560
 CF_CFDP_PduMd, 563
 CF_Chunk, 577
 CF_Logical_PduEof, 615
 CF_Logical_PduMd, 622
 source_eid
 CF_Logical_PduHeader, 620
 source_filename
 CF_Logical_PduMd, 622
 SpacecraftID
 CFE_EVS_PacketID, 723
 CFE_FS_Header, 734
 Spare
 CFE_ES_TaskInfo, 702
 CFE_EVS_AppNameBitMaskCmd_Payload, 706
 CFE_EVS_BitMaskCmd_Payload, 710
 CFE_EVS_SetEventFormatCode_Payload, 726
 CFE_EVS_SetLogMode_Payload, 728
 CFE_SB_PipeDepthStats, 745
 CFE_SB_PipeInfoEntry, 747
 CFE_SB_RouteCmd_Payload, 749
 spare

CF_HkChannel_Data, 601
 CF_HkFault, 605
 CF_HkPacket_Payload, 607
 CF_SetParam_Payload, 638
 CF_Transaction_Payload, 647
 CF_WriteQueue_Payload, 658
 Spare1
 CFE_EVS_HousekeepingTlm_Payload, 719
 CFE_EVS_LongEventTlm_Payload, 721
 Spare2
 CFE_EVS_HousekeepingTlm_Payload, 719
 CFE_EVS_LongEventTlm_Payload, 721
 Spare2Align
 CFE_SB_HousekeepingTlm_Payload, 741
 Spare3
 CFE_EVS_HousekeepingTlm_Payload, 719
 spurious
 CF_HkRecv, 608
 src_dir
 CF_PollDir, 633
 src_eid
 CF_EotPacket_Payload, 592
 CF_History, 599
 CF_Traverse_TransSeqArg, 649
 src_filename
 CF_TxFile_Payload, 653
 CF_TxnFilenames, 654
 stack_size
 OS_task_prop_t, 826
 StackSize
 CFE_ES_AppInfo, 664
 CFE_ES_StartAppCmd_Payload, 698
 CFE_ES_TaskInfo, 703
 start_time
 OS_timer_prop_t, 828
 StartAddress
 CFE_ES_AppInfo, 664
 State
 CFE_SB_RoutingFileEntry, 750
 state
 CF_CFDP_R_SubstateDispatchTable_t, 564
 CF_EotPacket_Payload, 592
 CF_Transaction, 646
 state_data
 CF_Transaction, 646
 StreamId
 CCSDS_PrimaryHeader, 554
 sub_state
 CF_RxState_Data, 637
 CF_TxState_Data, 655
 SubscribeErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 741
 SubscriptionsInUse
 CFE_SB_StatsTlm_Payload, 756

Subseconds
 CFE_TIME_OneHzAdjustmentCmd_Payload, 800
 CFE_TIME_SysTime, 810

Subsecs1HzAdj
 CFE_TIME_HousekeepingTlm_Payload, 798

SubsecsDelay
 CFE_TIME_HousekeepingTlm_Payload, 798

SubsecsMET
 CFE_TIME_HousekeepingTlm_Payload, 798

SubsecsSTCF
 CFE_TIME_HousekeepingTlm_Payload, 799

substate
 CF_CFDP_S_SubstateRecvDispatchTable_t, 565
 CF_CFDP_S_SubstateSendDispatchTable_t, 565

Subsystem
 CCSDS_ExtendedHeader, 553

SubType
 CFE_FS_Header, 734
 CFE_SB_SingleSubscriptionTlm_Payload, 753

SuccessValCounter
 CFE_TBL_HousekeepingTlm_Payload, 772

suspended
 CF_Flags_Common, 593

SysLogBytesUsed
 CFE_ES_HousekeepingTlm_Payload, 680

SysLogEntries
 CFE_ES_HousekeepingTlm_Payload, 680

SysLogMode
 CFE_ES_HousekeepingTlm_Payload, 680

SysLogSize
 CFE_ES_HousekeepingTlm_Payload, 680

SystemId
 CCSDS_ExtendedHeader, 553

Table
 CFE_ES_CDSRegDumpRec, 668

TableLoadedOnce
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 783

TableName
 CFE_TBL_AbortLoadCmd_Payload, 761
 CFE_TBL_ActivateCmd_Payload, 762
 CFE_TBL_DelCDSCmd_Payload, 762
 CFE_TBL_DumpCmd_Payload, 764
 CFE_TBL_File_Hdr, 766
 CFE_TBL_FileDef, 768
 CFE_TBL_SendRegistryCmd_Payload, 779
 CFE_TBL_ValidateCmd_Payload, 784

TaskId
 CFE_ES_TaskInfo, 703

TaskName
 CFE_ES_TaskInfo, 703

TelemetryHeader
 CF_EotPacket, 590

CF_HkPacket, 606

CFE_ES_HousekeepingTlm, 672

CFE_ES_MemStatsTlm, 682

CFE_ES_OneAppTlm, 683

CFE_EVS_HousekeepingTlm, 716

CFE_EVS_LongEventTlm, 720

CFE_EVS_ShortEventTlm, 729

CFE_SB_AllSubscriptionsTlm, 735

CFE_SB_HousekeepingTlm, 738

CFE_SB_SingleSubscriptionTlm, 752

CFE_SB_StatsTlm, 753

CFE_TBL_HousekeepingTlm, 768

CFE_TBL_TableRegistryTlm, 780

CFE_TIME_DiagnosticTlm, 787

CFE_TIME_HousekeepingTlm, 796

TgtFilename
 CFE_TBL_FileDef, 768

tick
 CF_Timer, 643

tick_type
 CF_Channel, 574

ticks
 OS_time_t, 826

ticks_per_second
 CF_ConfigTable, 582

TimeOfLastUpdate
 CFE_TBL_Info, 774
 CFE_TBL_TblRegPacket_Payload, 783

timer_set
 CF_Poll, 632

TimeSeconds
 CFE_FS_Header, 734

TimeSinceTone
 CFE_TIME_DiagnosticTlm_Payload, 793

TimeSource
 CFE_TIME_SourceCmd_Payload, 807

TimeSubSeconds
 CFE_FS_Header, 734

tlv
 CF_Logical_TlvList, 626

tlv_list
 CF_Logical_PduEof, 615
 CF_Logical_PduFin, 618

tmp_dir
 CF_ConfigTable, 582

ToneDataCounter
 CFE_TIME_DiagnosticTlm_Payload, 793

ToneDataLatch
 CFE_TIME_DiagnosticTlm_Payload, 793

ToneIntCounter
 CFE_TIME_DiagnosticTlm_Payload, 793

ToneIntErrorCounter
 CFE_TIME_DiagnosticTlm_Payload, 793

ToneMatchCounter

CFE_TIME_DiagnosticTlm_Payload, 794
 ToneMatchErrorCounter
 CFE_TIME_DiagnosticTlm_Payload, 794
 ToneOverLimit
 CFE_TIME_DiagnosticTlm_Payload, 794
 ToneSignalCounter
 CFE_TIME_DiagnosticTlm_Payload, 794
 ToneSignalLatch
 CFE_TIME_DiagnosticTlm_Payload, 794
 ToneSource
 CFE_TIME_SignalCmd_Payload, 807
 ToneTaskCounter
 CFE_TIME_DiagnosticTlm_Payload, 794
 ToneUnderLimit
 CFE_TIME_DiagnosticTlm_Payload, 794
 total_blocks
 OS_statvfs_t, 825
 TotalSegments
 CFE_SB_AllSubscriptionsTlm_Payload, 736
 transaction_sequence_number
 CF_Traverse_TransSeqArg, 649
 transactions
 CF_Engine, 589
 TriggerMask
 CFE_ES_SetPerfTrigMaskCmd_Payload, 696
 TriggerMaskNum
 CFE_ES_SetPerfTrigMaskCmd_Payload, 697
 TriggerMode
 CFE_ES_StartPerfCmd_Payload, 699
 ts
 CF_Transaction_Payload, 647
 tx
 CF_CFDP_TxnSendDispatchTable_t, 568
 CF_StateFlags, 641
 tx_pudata
 CF_Output, 627
 txm_mode
 CF_Logical_PduHeader, 620
 txn
 CF_GapComputeArgs_t, 597
 CF_Traverse_PriorityArg, 648
 CF_Traverse_TransSeqArg, 649
 txn_seq_length
 CF_Logical_PduHeader, 620
 txn_stat
 CF_EotPacket_Payload, 592
 CF_History, 600
 txn_status
 CF_Logical_PduAck, 613
 Type
 CFE_ES_AppInfo, 664
 type
 CF_CFDP_tlv, 567
 CF_Logical_Tlv, 625
 CF_WriteQueue_Payload, 658
 uint16
 common_types.h, 1701
 uint32
 common_types.h, 1701
 uint64
 common_types.h, 1701
 uint8
 common_types.h, 1701
 UnmarkedMem
 CFE_SB_HousekeepingTlm_Payload, 741
 UnregisteredAppCounter
 CFE_EVS_HousekeepingTlm_Payload, 719
 User
 OS_file_prop_t, 816
 UserDefAddr
 CFE_TBL_Info, 774
 valid
 OS_module_address_t, 820
 ValidationCounter
 CFE_TBL_HousekeepingTlm_Payload, 772
 ValidationFuncPtr
 CFE_TBL_TblRegPacket_Payload, 783
 Value
 CFE_SB_MsgId_t, 743
 value
 CF_SetParam_Payload, 638
 OS_bin_sem_prop_t, 813
 OS_count_sem_prop_t, 815
 version
 CF_Logical_PduHeader, 621
 VersionCounter
 CFE_TIME_DiagnosticTlm_Payload, 795
 VirtualMET
 CFE_TIME_DiagnosticTlm_Payload, 795
 working
 CF_Crc, 583