

HS User's Guide

Generated by Doxygen 1.8.13

Contents

1 CFS Health and Safety (HS) Documentation	1
1.1 CFS Health and Safety Introduction	2
1.2 CFS Health and Safety Overview	3
1.2.1 Design Overview	3
1.3 CFS Health and Safety Operation	3
1.4 CFS Health and Safety Commands	6
1.5 CFS Health and Safety Telemetry	6
1.6 CFS Health and Safety Events	6
1.7 CFS Health and Safety Deployment Guide	6
1.8 CFS Health and Safety Configuration	8
1.9 CFS Health and Safety Table Definitions	9
1.10 CFS Health and Safety Operational Constraints	11
1.11 CFS Health and Safety Frequently Asked Questions	11
2 Core Flight Executive Documentation	11
2.1 Background	12
2.2 Applicable Documents	14
2.3 Version Numbers	14
2.4 Dependencies	16
2.5 Acronyms	16
2.6 cFE Executive Services Overview	17
2.6.1 Terminology	18
2.6.2 Software Reset	20
2.6.3 Reset Types and Subtypes	21
2.6.4 Exception and Reset (ER) Log	21
2.6.5 Application and Child Task Management	21
2.6.6 Starting an Application	22

2.6.7 Stopping an Application	22
2.6.8 Restarting an Application	23
2.6.9 Reloading an Application	23
2.6.10 Listing Current Applications	23
2.6.11 Listing Current Tasks	24
2.6.12 Loading Common Libraries	25
2.6.13 Basic File System	25
2.6.14 Performance Data Collection	25
2.6.15 Critical Data Store	27
2.6.16 Memory Pool	27
2.6.17 System Log	30
2.6.18 Version Identification	30
2.6.19 Executive Services Frequently Asked Questions	30
2.7 cFE Executive Services Commands	30
2.8 cFE Executive Services Telemetry	32
2.9 cFE Executive Services Configuration Parameters	32
2.10 cFE Event Services Overview	35
2.10.1 Event Message Format	36
2.10.2 Local Event Log	37
2.10.3 Event Message Control	38
2.10.4 Event Message Filtering	40
2.10.5 EVS Registry	40
2.10.6 EVS Counters	41
2.10.7 Resetting EVS Counters	42
2.10.8 Effects of a Processor Reset on EVS	42
2.10.9 Frequently Asked Questions about Event Services	42
2.11 cFE Event Services Commands	43
2.12 cFE Event Services Telemetry	45

2.13 cFE Event Services Configuration Parameters	45
2.14 cFE Software Bus Overview	46
2.14.1 Software Bus Terminology	46
2.14.2 Autonomous Actions	49
2.14.3 Operation of the SB Software	49
2.14.4 Frequently Asked Questions about Software Bus	53
2.15 cFE Software Bus Commands	55
2.16 cFE Software Bus Telemetry	56
2.17 cFE Software Bus Configuration Parameters	56
2.18 cFE Table Services Overview	57
2.18.1 Managing Tables	58
2.18.2 cFE Table Types and Table Options	59
2.18.3 Table Registry	61
2.18.4 Table Services Telemetry	62
2.18.5 Effects of Processor Reset on Tables	62
2.18.6 Frequently Asked Questions about Table Services	62
2.19 cFE Table Services Commands	64
2.20 cFE Table Services Telemetry	65
2.21 cFE Table Services Configuration Parameters	65
2.22 cFE Time Services Overview	66
2.22.1 Time Components	68
2.22.2 Time Structure	69
2.22.3 Time Formats	69
2.22.4 Time Configuration	70
2.22.5 Time Format Selection	75
2.22.6 Enabling Fake Tone Signal	76
2.22.7 Selecting Tone and Data Ordering	76
2.22.8 Specifying Tone and Data Window	77

2.22.9 Specifying Time Server/Client	77
2.22.10 Specifying Time Tone Byte Order	78
2.22.11 Virtual MET	78
2.22.12 Specifying Time Source	78
2.22.13 Specifying Time Signal	79
2.22.14 Time Services Paradigm(s)	80
2.22.15 Flywheeling	80
2.22.16 Time State	81
2.22.17 Initialization	81
2.22.18 Power-On Reset	82
2.22.19 Processor Reset	83
2.22.20 Initialization	83
2.22.21 Power-On Reset	84
2.22.22 Processor Reset	85
2.22.23 Normal Operation	85
2.22.24 Client	87
2.22.25 Server	88
2.22.26 Setting Time	89
2.22.27 Adjusting Time	90
2.22.28 Setting MET	90
2.22.29 Frequently Asked Questions	90
2.23 cFE Time Services Commands	91
2.24 cFE Time Services Telemetry	92
2.25 cFE Time Services Configuration Parameters	92
2.26 cFE Event Message Cross Reference	93
2.27 cFE Command Mnemonic Cross Reference	93
2.28 cFE Telemetry Mnemonic Cross Reference	97

3	Glossary of Terms	108
4	cFE Application Programmer's Interface (API) Reference	109
5	Osal API Documentation	116
5.1	OSAL Introduction	118
5.2	File System Overview	118
5.3	File Descriptors In Osal	119
5.4	Timer Overview	120
6	cFE Mission Configuration Parameters	120
7	Module Index	120
7.1	Modules	120
8	Data Structure Index	123
8.1	Data Structures	123
9	File Index	131
9.1	File List	132
10	Module Documentation	135
10.1	CFS Health and Safety Mission Configuration	135
10.1.1	Detailed Description	135
10.1.2	Macro Definition Documentation	135
10.2	CFS Health and Safety Command Message IDs	137
10.2.1	Detailed Description	137
10.2.2	Macro Definition Documentation	137
10.3	CFS Health and Safety Telemetry Message IDs	138
10.3.1	Detailed Description	138
10.3.2	Macro Definition Documentation	138
10.4	CFS Health and Safety Platform Configuration	139

10.4.1 Detailed Description	140
10.4.2 Macro Definition Documentation	140
10.5 CFS Health and Safety Event IDs	159
10.5.1 Detailed Description	162
10.5.2 Macro Definition Documentation	162
10.6 CFS Health and Safety Command Structures	204
10.6.1 Detailed Description	204
10.7 CFS Health and Safety Telemetry	205
10.7.1 Detailed Description	205
10.8 CFS Health and Safety Command Codes	206
10.8.1 Detailed Description	206
10.8.2 Macro Definition Documentation	206
10.9 CFS Health and Safety Version	222
10.9.1 Detailed Description	222
10.9.2 Macro Definition Documentation	222
10.10cFE Return Code Defines	223
10.10.1 Detailed Description	228
10.10.2 Macro Definition Documentation	228
10.11cFE Resource ID APIs	260
10.11.1 Detailed Description	260
10.11.2 Function Documentation	260
10.12cFE Entry/Exit APIs	264
10.12.1 Detailed Description	264
10.12.2 Function Documentation	264
10.13cFE Application Control APIs	266
10.13.1 Detailed Description	266
10.13.2 Function Documentation	266
10.14cFE Application Behavior APIs	269

10.14.1 Detailed Description	269
10.14.2 Function Documentation	269
10.15cFE Information APIs	274
10.15.1 Detailed Description	274
10.15.2 Function Documentation	274
10.16cFE Child Task APIs	284
10.16.1 Detailed Description	284
10.16.2 Function Documentation	284
10.17cFE Miscellaneous APIs	289
10.17.1 Detailed Description	289
10.17.2 Function Documentation	289
10.18cFE Critical Data Store APIs	293
10.18.1 Detailed Description	293
10.18.2 Function Documentation	293
10.19cFE Memory Manager APIs	299
10.19.1 Detailed Description	299
10.19.2 Function Documentation	299
10.20cFE Performance Monitor APIs	307
10.20.1 Detailed Description	307
10.20.2 Macro Definition Documentation	307
10.20.3 Function Documentation	308
10.21cFE Generic Counter APIs	310
10.21.1 Detailed Description	310
10.21.2 Function Documentation	310
10.22cFE Registration APIs	317
10.22.1 Detailed Description	317
10.22.2 Function Documentation	317
10.23cFE Send Event APIs	319

10.23.1 Detailed Description	319
10.23.2 Function Documentation	319
10.24cFE Reset Event Filter APIs	324
10.24.1 Detailed Description	324
10.24.2 Function Documentation	324
10.25cFE File Header Management APIs	326
10.25.1 Detailed Description	326
10.25.2 Function Documentation	326
10.26cFE File Utility APIs	331
10.26.1 Detailed Description	331
10.26.2 Function Documentation	331
10.27cFE Generic Message APIs	337
10.27.1 Detailed Description	337
10.27.2 Function Documentation	337
10.28cFE Message Primary Header APIs	338
10.28.1 Detailed Description	338
10.28.2 Function Documentation	338
10.29cFE Message Extended Header APIs	349
10.29.1 Detailed Description	349
10.29.2 Function Documentation	349
10.30cFE Message Secondary Header APIs	356
10.30.1 Detailed Description	356
10.30.2 Function Documentation	356
10.31cFE Message Id APIs	362
10.31.1 Detailed Description	362
10.31.2 Function Documentation	362
10.32cFE Pipe Management APIs	365
10.32.1 Detailed Description	365

10.32.2 Function Documentation	365
10.33cFE Message Subscription Control APIs	372
10.33.1 Detailed Description	372
10.33.2 Function Documentation	372
10.34cFE Send/Receive Message APIs	378
10.34.1 Detailed Description	378
10.34.2 Function Documentation	378
10.35cFE Zero Copy APIs	381
10.35.1 Detailed Description	381
10.35.2 Function Documentation	381
10.36cFE Message Characteristics APIs	384
10.36.1 Detailed Description	384
10.36.2 Function Documentation	384
10.37cFE Message ID APIs	389
10.37.1 Detailed Description	389
10.37.2 Function Documentation	389
10.38cFE SB Pipe options	392
10.38.1 Detailed Description	392
10.38.2 Macro Definition Documentation	392
10.39cFE Registration APIs	393
10.39.1 Detailed Description	393
10.39.2 Function Documentation	393
10.40cFE Manage Table Content APIs	399
10.40.1 Detailed Description	399
10.40.2 Function Documentation	399
10.41cFE Access Table Content APIs	406
10.41.1 Detailed Description	406
10.41.2 Function Documentation	406

10.42cFE Get Table Information APIs	411
10.42.1 Detailed Description	411
10.42.2 Function Documentation	411
10.43cFE Table Type Defines	415
10.43.1 Detailed Description	415
10.43.2 Macro Definition Documentation	415
10.44cFE Get Current Time APIs	418
10.44.1 Detailed Description	418
10.44.2 Function Documentation	418
10.45cFE Get Time Information APIs	422
10.45.1 Detailed Description	422
10.45.2 Function Documentation	422
10.46cFE Time Arithmetic APIs	425
10.46.1 Detailed Description	425
10.46.2 Function Documentation	425
10.47cFE Time Conversion APIs	428
10.47.1 Detailed Description	428
10.47.2 Function Documentation	428
10.48cFE External Time Source APIs	431
10.48.1 Detailed Description	431
10.48.2 Function Documentation	431
10.49cFE Miscellaneous Time APIs	436
10.49.1 Detailed Description	436
10.49.2 Function Documentation	436
10.50cFE Resource ID base values	438
10.50.1 Detailed Description	438
10.50.2 Enumeration Type Documentation	438
10.51cFE Clock State Flag Defines	440

10.51.1 Detailed Description	440
10.51.2 Macro Definition Documentation	440
10.52OSAL Semaphore State Defines	443
10.52.1 Detailed Description	443
10.52.2 Macro Definition Documentation	443
10.53OSAL Binary Semaphore APIs	444
10.53.1 Detailed Description	444
10.53.2 Function Documentation	444
10.54OSAL BSP low level access APIs	450
10.54.1 Detailed Description	450
10.54.2 Function Documentation	450
10.55OSAL Real Time Clock APIs	451
10.55.1 Detailed Description	451
10.55.2 Function Documentation	451
10.56OSAL Core Operation APIs	463
10.56.1 Detailed Description	463
10.56.2 Function Documentation	463
10.57OSAL Counting Semaphore APIs	467
10.57.1 Detailed Description	467
10.57.2 Function Documentation	467
10.58OSAL Directory APIs	473
10.58.1 Detailed Description	473
10.58.2 Function Documentation	473
10.59OSAL Return Code Defines	478
10.59.1 Detailed Description	479
10.59.2 Macro Definition Documentation	480
10.60OSAL Error Info APIs	489
10.60.1 Detailed Description	489

10.60.2 Function Documentation	489
10.61OSAL File Access Option Defines	491
10.61.1 Detailed Description	491
10.61.2 Macro Definition Documentation	491
10.62OSAL Reference Point For Seek Offset Defines	492
10.62.1 Detailed Description	492
10.62.2 Macro Definition Documentation	492
10.63OSAL Standard File APIs	493
10.63.1 Detailed Description	493
10.63.2 Function Documentation	493
10.64OSAL File System Level APIs	506
10.64.1 Detailed Description	506
10.64.2 Function Documentation	506
10.65OSAL Heap APIs	515
10.65.1 Detailed Description	515
10.65.2 Function Documentation	515
10.66OSAL Object Type Defines	516
10.66.1 Detailed Description	516
10.66.2 Macro Definition Documentation	516
10.67OSAL Object ID Utility APIs	520
10.67.1 Detailed Description	520
10.67.2 Function Documentation	520
10.68OSAL Dynamic Loader and Symbol APIs	527
10.68.1 Detailed Description	527
10.68.2 Function Documentation	527
10.69OSAL Mutex APIs	532
10.69.1 Detailed Description	532
10.69.2 Function Documentation	532

10.70OSAL Network ID APIs	537
10.70.1 Detailed Description	537
10.70.2 Function Documentation	537
10.71OSAL Printf APIs	539
10.71.1 Detailed Description	539
10.71.2 Function Documentation	539
10.72OSAL Message Queue APIs	541
10.72.1 Detailed Description	541
10.72.2 Function Documentation	541
10.73OSAL Select APIs	546
10.73.1 Detailed Description	546
10.73.2 Function Documentation	546
10.74OSAL Shell APIs	551
10.74.1 Detailed Description	551
10.74.2 Function Documentation	551
10.75OSAL Socket Address APIs	552
10.75.1 Detailed Description	552
10.75.2 Function Documentation	552
10.76OSAL Socket Management APIs	556
10.76.1 Detailed Description	556
10.76.2 Function Documentation	556
10.77OSAL Task APIs	564
10.77.1 Detailed Description	564
10.77.2 Function Documentation	564
10.78OSAL Time Base APIs	571
10.78.1 Detailed Description	571
10.78.2 Function Documentation	571
10.79OSAL Timer APIs	577
10.79.1 Detailed Description	577
10.79.2 Function Documentation	577

11 Data Structure Documentation	584
11.1 CCSDS_ExtendedHeader Struct Reference	584
11.1.1 Detailed Description	584
11.1.2 Field Documentation	584
11.2 CCSDS_PrimaryHeader Struct Reference	585
11.2.1 Detailed Description	585
11.2.2 Field Documentation	585
11.3 CFE_ES_AppInfo Struct Reference	586
11.3.1 Detailed Description	587
11.3.2 Field Documentation	587
11.4 CFE_ES_AppNameCmd Struct Reference	592
11.4.1 Detailed Description	593
11.4.2 Field Documentation	593
11.5 CFE_ES_AppNameCmd_Payload Struct Reference	593
11.5.1 Detailed Description	594
11.5.2 Field Documentation	594
11.6 CFE_ES_AppReloadCmd_Payload Struct Reference	594
11.6.1 Detailed Description	595
11.6.2 Field Documentation	595
11.7 CFE_ES_BlockStats Struct Reference	595
11.7.1 Detailed Description	596
11.7.2 Field Documentation	596
11.8 CFE_ES_CDSRegDumpRec Struct Reference	596
11.8.1 Detailed Description	597
11.8.2 Field Documentation	597
11.9 CFE_ES_DeleteCDSCmd Struct Reference	598
11.9.1 Detailed Description	599
11.9.2 Field Documentation	599

11.10CFE_ES_DeleteCDSCmd_Payload Struct Reference	599
11.10.1 Detailed Description	600
11.10.2 Field Documentation	600
11.11CFE_ES_DumpCDSRegistryCmd Struct Reference	600
11.11.1 Detailed Description	600
11.11.2 Field Documentation	601
11.12CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference	601
11.12.1 Detailed Description	601
11.12.2 Field Documentation	602
11.13CFE_ES_FileNameCmd Struct Reference	602
11.13.1 Detailed Description	602
11.13.2 Field Documentation	602
11.14CFE_ES_FileNameCmd_Payload Struct Reference	603
11.14.1 Detailed Description	603
11.14.2 Field Documentation	603
11.15CFE_ES_HousekeepingTlm Struct Reference	604
11.15.1 Detailed Description	604
11.15.2 Field Documentation	604
11.16CFE_ES_HousekeepingTlm_Payload Struct Reference	604
11.16.1 Detailed Description	606
11.16.2 Field Documentation	607
11.17CFE_ES_MemPoolStats Struct Reference	617
11.17.1 Detailed Description	618
11.17.2 Field Documentation	618
11.18CFE_ES_MemStatsTlm Struct Reference	619
11.18.1 Detailed Description	620
11.18.2 Field Documentation	620
11.19CFE_ES_NoArgsCmd Struct Reference	620

11.19.1 Detailed Description	621
11.19.2 Field Documentation	621
11.20CFE_ES_OneAppTlm Struct Reference	621
11.20.1 Detailed Description	622
11.20.2 Field Documentation	622
11.21CFE_ES_OneAppTlm_Payload Struct Reference	622
11.21.1 Detailed Description	622
11.21.2 Field Documentation	623
11.22CFE_ES_OverWriteSysLogCmd Struct Reference	623
11.22.1 Detailed Description	623
11.22.2 Field Documentation	623
11.23CFE_ES_OverWriteSysLogCmd_Payload Struct Reference	624
11.23.1 Detailed Description	624
11.23.2 Field Documentation	624
11.24CFE_ES_PoolAlign Union Reference	625
11.24.1 Detailed Description	625
11.24.2 Field Documentation	625
11.25CFE_ES_PoolStatsTlm_Payload Struct Reference	626
11.25.1 Detailed Description	626
11.25.2 Field Documentation	626
11.26CFE_ES_ReloadAppCmd Struct Reference	627
11.26.1 Detailed Description	627
11.26.2 Field Documentation	627
11.27CFE_ES_RestartCmd Struct Reference	628
11.27.1 Detailed Description	628
11.27.2 Field Documentation	628
11.28CFE_ES_RestartCmd_Payload Struct Reference	629
11.28.1 Detailed Description	629

11.28.2 Field Documentation	629
11.29 CFE_ES_SendMemPoolStatsCmd Struct Reference	630
11.29.1 Detailed Description	630
11.29.2 Field Documentation	630
11.30 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference	631
11.30.1 Detailed Description	631
11.30.2 Field Documentation	631
11.31 CFE_ES_SetMaxPRCountCmd Struct Reference	632
11.31.1 Detailed Description	632
11.31.2 Field Documentation	632
11.32 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference	633
11.32.1 Detailed Description	633
11.32.2 Field Documentation	633
11.33 CFE_ES_SetPerfFilterMaskCmd Struct Reference	633
11.33.1 Detailed Description	634
11.33.2 Field Documentation	634
11.34 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference	634
11.34.1 Detailed Description	635
11.34.2 Field Documentation	635
11.35 CFE_ES_SetPerfTriggerMaskCmd Struct Reference	635
11.35.1 Detailed Description	636
11.35.2 Field Documentation	636
11.36 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference	636
11.36.1 Detailed Description	637
11.36.2 Field Documentation	637
11.37 CFE_ES_StartApp Struct Reference	637
11.37.1 Detailed Description	638
11.37.2 Field Documentation	638

11.38CFE_ES_StartAppCmd_Payload Struct Reference	638
11.38.1 Detailed Description	639
11.38.2 Field Documentation	639
11.39CFE_ES_StartPerfCmd_Payload Struct Reference	640
11.39.1 Detailed Description	641
11.39.2 Field Documentation	641
11.40CFE_ES_StartPerfDataCmd Struct Reference	641
11.40.1 Detailed Description	642
11.40.2 Field Documentation	642
11.41CFE_ES_StopPerfCmd_Payload Struct Reference	642
11.41.1 Detailed Description	643
11.41.2 Field Documentation	643
11.42CFE_ES_StopPerfDataCmd Struct Reference	643
11.42.1 Detailed Description	643
11.42.2 Field Documentation	644
11.43CFE_ES_TaskInfo Struct Reference	644
11.43.1 Detailed Description	645
11.43.2 Field Documentation	645
11.44CFE_EVS_AppDataCmd_Payload Struct Reference	647
11.44.1 Detailed Description	647
11.44.2 Field Documentation	647
11.45CFE_EVS_AppNameBitMaskCmd Struct Reference	647
11.45.1 Detailed Description	648
11.45.2 Field Documentation	648
11.46CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference	648
11.46.1 Detailed Description	649
11.46.2 Field Documentation	649
11.47CFE_EVS_AppNameCmd Struct Reference	650

11.47.1 Detailed Description	650
11.47.2 Field Documentation	650
11.48CFE_EVS_AppNameCmd_Payload Struct Reference	651
11.48.1 Detailed Description	651
11.48.2 Field Documentation	651
11.49CFE_EVS_AppNameEventIDCmd Struct Reference	652
11.49.1 Detailed Description	652
11.49.2 Field Documentation	652
11.50CFE_EVS_AppNameEventIDCmd_Payload Struct Reference	653
11.50.1 Detailed Description	653
11.50.2 Field Documentation	653
11.51CFE_EVS_AppNameEventIDMaskCmd Struct Reference	654
11.51.1 Detailed Description	654
11.51.2 Field Documentation	654
11.52CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference	655
11.52.1 Detailed Description	655
11.52.2 Field Documentation	655
11.53CFE_EVS_AppTlmData Struct Reference	656
11.53.1 Detailed Description	656
11.53.2 Field Documentation	656
11.54CFE_EVS_BinFilter Struct Reference	657
11.54.1 Detailed Description	658
11.54.2 Field Documentation	658
11.55CFE_EVS_BitMaskCmd Struct Reference	658
11.55.1 Detailed Description	659
11.55.2 Field Documentation	659
11.56CFE_EVS_BitMaskCmd_Payload Struct Reference	659
11.56.1 Detailed Description	660

11.56.2 Field Documentation	660
11.57CFE_EVS_HousekeepingTlm Struct Reference	660
11.57.1 Detailed Description	661
11.57.2 Field Documentation	661
11.58CFE_EVS_HousekeepingTlm_Payload Struct Reference	661
11.58.1 Detailed Description	662
11.58.2 Field Documentation	662
11.59CFE_EVS_LogFileCmd_Payload Struct Reference	666
11.59.1 Detailed Description	667
11.59.2 Field Documentation	667
11.60CFE_EVS_LongEventTlm Struct Reference	667
11.60.1 Detailed Description	667
11.60.2 Field Documentation	668
11.61CFE_EVS_LongEventTlm_Payload Struct Reference	668
11.61.1 Detailed Description	668
11.61.2 Field Documentation	669
11.62CFE_EVS_NoArgsCmd Struct Reference	670
11.62.1 Detailed Description	670
11.62.2 Field Documentation	670
11.63CFE_EVS_PacketID Struct Reference	670
11.63.1 Detailed Description	671
11.63.2 Field Documentation	671
11.64CFE_EVS_SetEventFormatCode_Payload Struct Reference	672
11.64.1 Detailed Description	672
11.64.2 Field Documentation	673
11.65CFE_EVS_SetEventFormatModeCmd Struct Reference	673
11.65.1 Detailed Description	673
11.65.2 Field Documentation	674

11.66CFE_EVS_SetLogMode_Payload Struct Reference	674
11.66.1 Detailed Description	674
11.66.2 Field Documentation	675
11.67CFE_EVS_SetLogModeCmd Struct Reference	675
11.67.1 Detailed Description	675
11.67.2 Field Documentation	676
11.68CFE_EVS_ShortEventTlm Struct Reference	676
11.68.1 Detailed Description	676
11.68.2 Field Documentation	676
11.69CFE_EVS_ShortEventTlm_Payload Struct Reference	677
11.69.1 Detailed Description	677
11.69.2 Field Documentation	677
11.70CFE_EVS_WriteAppDataFileCmd Struct Reference	678
11.70.1 Detailed Description	678
11.70.2 Field Documentation	678
11.71CFE_EVS_WriteLogFileCmd Struct Reference	679
11.71.1 Detailed Description	679
11.71.2 Field Documentation	679
11.72CFE_FS_FileWriteMetaData Struct Reference	680
11.72.1 Detailed Description	680
11.72.2 Field Documentation	680
11.73CFE_FS_Header Struct Reference	682
11.73.1 Detailed Description	682
11.73.2 Field Documentation	682
11.74CFE_SB_AllSubscriptionsTlm Struct Reference	684
11.74.1 Detailed Description	685
11.74.2 Field Documentation	685
11.75CFE_SB_AllSubscriptionsTlm_Payload Struct Reference	685

11.75.1 Detailed Description	686
11.75.2 Field Documentation	686
11.76CFE_SB_HousekeepingTlm Struct Reference	687
11.76.1 Detailed Description	687
11.76.2 Field Documentation	687
11.77CFE_SB_HousekeepingTlm_Payload Struct Reference	688
11.77.1 Detailed Description	689
11.77.2 Field Documentation	689
11.78CFE_SB_Msg Union Reference	693
11.78.1 Detailed Description	693
11.78.2 Field Documentation	694
11.79CFE_SB_MsgId_t Struct Reference	694
11.79.1 Detailed Description	695
11.79.2 Field Documentation	695
11.80CFE_SB_MsgMapFileEntry Struct Reference	695
11.80.1 Detailed Description	696
11.80.2 Field Documentation	696
11.81CFE_SB_PipeDepthStats Struct Reference	696
11.81.1 Detailed Description	697
11.81.2 Field Documentation	697
11.82CFE_SB_PipeInfoEntry Struct Reference	698
11.82.1 Detailed Description	699
11.82.2 Field Documentation	699
11.83CFE_SB_Qos_t Struct Reference	701
11.83.1 Detailed Description	702
11.83.2 Field Documentation	702
11.84CFE_SB_RouteCmd Struct Reference	702
11.84.1 Detailed Description	703

11.84.2 Field Documentation	703
11.85CFE_SB_RouteCmd_Payload Struct Reference	703
11.85.1 Detailed Description	704
11.85.2 Field Documentation	704
11.86CFE_SB_RoutingFileEntry Struct Reference	705
11.86.1 Detailed Description	705
11.86.2 Field Documentation	705
11.87CFE_SB_SingleSubscriptionTlm Struct Reference	707
11.87.1 Detailed Description	707
11.87.2 Field Documentation	707
11.88CFE_SB_SingleSubscriptionTlm_Payload Struct Reference	707
11.88.1 Detailed Description	708
11.88.2 Field Documentation	708
11.89CFE_SB_StatsTlm Struct Reference	709
11.89.1 Detailed Description	709
11.89.2 Field Documentation	709
11.90CFE_SB_StatsTlm_Payload Struct Reference	710
11.90.1 Detailed Description	711
11.90.2 Field Documentation	711
11.91CFE_SB_SubEntries Struct Reference	715
11.91.1 Detailed Description	716
11.91.2 Field Documentation	716
11.92CFE_SB_WriteFileInfoCmd Struct Reference	717
11.92.1 Detailed Description	717
11.92.2 Field Documentation	717
11.93CFE_SB_WriteFileInfoCmd_Payload Struct Reference	718
11.93.1 Detailed Description	718
11.93.2 Field Documentation	718

11.94CFE_TBL_AbortLoadCmd Struct Reference	718
11.94.1 Detailed Description	719
11.94.2 Field Documentation	719
11.95CFE_TBL_AbortLoadCmd_Payload Struct Reference	719
11.95.1 Detailed Description	720
11.95.2 Field Documentation	720
11.96CFE_TBL_ActivateCmd Struct Reference	720
11.96.1 Detailed Description	721
11.96.2 Field Documentation	721
11.97CFE_TBL_ActivateCmd_Payload Struct Reference	721
11.97.1 Detailed Description	722
11.97.2 Field Documentation	722
11.98CFE_TBL_DelCDSCmd_Payload Struct Reference	722
11.98.1 Detailed Description	722
11.98.2 Field Documentation	723
11.99CFE_TBL_DeleteCDSCmd Struct Reference	723
11.99.1 Detailed Description	723
11.99.2 Field Documentation	723
11.10CFE_TBL_DumpCmd Struct Reference	724
11.100.1 Detailed Description	724
11.100.2 Field Documentation	724
11.10CFE_TBL_DumpCmd_Payload Struct Reference	725
11.101.1 Detailed Description	725
11.101.2 Field Documentation	726
11.10CFE_TBL_DumpRegistryCmd Struct Reference	726
11.102.1 Detailed Description	727
11.102.2 Field Documentation	727
11.10CFE_TBL_DumpRegistryCmd_Payload Struct Reference	727

11.103.1 Detailed Description	728
11.103.2 Field Documentation	728
11.104 CFE_TBL_File_Hdr Struct Reference	728
11.104.1 Detailed Description	729
11.104.2 Field Documentation	729
11.105 CFE_TBL_FileDef Struct Reference	730
11.105.1 Detailed Description	730
11.105.2 Field Documentation	730
11.106 CFE_TBL_HousekeepingTlm Struct Reference	731
11.106.1 Detailed Description	731
11.106.2 Field Documentation	732
11.107 CFE_TBL_HousekeepingTlm_Payload Struct Reference	732
11.107.1 Detailed Description	733
11.107.2 Field Documentation	733
11.108 CFE_TBL_Info Struct Reference	738
11.108.1 Detailed Description	739
11.108.2 Field Documentation	739
11.109 CFE_TBL_LoadCmd Struct Reference	742
11.109.1 Detailed Description	742
11.109.2 Field Documentation	742
11.110 CFE_TBL_LoadCmd_Payload Struct Reference	743
11.110.1 Detailed Description	743
11.110.2 Field Documentation	743
11.111 CFE_TBL_NoArgsCmd Struct Reference	744
11.111.1 Detailed Description	744
11.111.2 Field Documentation	744
11.112 CFE_TBL_NotifyCmd Struct Reference	744
11.112.1 Detailed Description	745

11.112. Field Documentation	745
11.113. CFE_TBL_NotifyCmd_Payload Struct Reference	745
11.113.1. Detailed Description	746
11.113.2. Field Documentation	746
11.114. CFE_TBL_SendRegistryCmd Struct Reference	746
11.114.1. Detailed Description	747
11.114.2. Field Documentation	747
11.115. CFE_TBL_SendRegistryCmd_Payload Struct Reference	747
11.115.1. Detailed Description	748
11.115.2. Field Documentation	748
11.116. CFE_TBL_TableRegistryTlm Struct Reference	748
11.116.1. Detailed Description	748
11.116.2. Field Documentation	749
11.117. CFE_TBL_TblRegPacket_Payload Struct Reference	749
11.117.1. Detailed Description	750
11.117.2. Field Documentation	750
11.118. CFE_TBL_ValidateCmd Struct Reference	755
11.118.1. Detailed Description	755
11.118.2. Field Documentation	755
11.119. CFE_TBL_ValidateCmd_Payload Struct Reference	756
11.119.1. Detailed Description	756
11.119.2. Field Documentation	756
11.120. CFE_TIME_DiagnosticTlm Struct Reference	757
11.120.1. Detailed Description	757
11.120.2. Field Documentation	757
11.121. CFE_TIME_DiagnosticTlm_Payload Struct Reference	757
11.121.1. Detailed Description	759
11.121.2. Field Documentation	760

11.12 CFE_TIME_HousekeepingTlm Struct Reference	770
11.122.1 Detailed Description	770
11.122.2 Field Documentation	771
11.12 CFE_TIME_HousekeepingTlm_Payload Struct Reference	771
11.123.1 Detailed Description	772
11.123.2 Field Documentation	772
11.12 CFE_TIME_LeapsCmd_Payload Struct Reference	775
11.124.1 Detailed Description	776
11.124.2 Field Documentation	776
11.12 CFE_TIME_NoArgsCmd Struct Reference	776
11.125.1 Detailed Description	776
11.125.2 Field Documentation	777
11.12 CFE_TIME_OneHzAdjustmentCmd Struct Reference	777
11.126.1 Detailed Description	777
11.126.2 Field Documentation	777
11.12 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference	778
11.127.1 Detailed Description	778
11.127.2 Field Documentation	778
11.12 CFE_TIME_SetLeapSecondsCmd Struct Reference	779
11.128.1 Detailed Description	779
11.128.2 Field Documentation	779
11.12 CFE_TIME_SetSignalCmd Struct Reference	780
11.129.1 Detailed Description	780
11.129.2 Field Documentation	780
11.13 CFE_TIME_SetSourceCmd Struct Reference	781
11.130.1 Detailed Description	781
11.130.2 Field Documentation	781
11.13 CFE_TIME_SetStateCmd Struct Reference	782

11.131.1. Detailed Description	782
11.131.2. Field Documentation	782
11.132. CFE_TIME_SignalCmd_Payload Struct Reference	783
11.132.1. Detailed Description	783
11.132.2. Field Documentation	783
11.133. CFE_TIME_SourceCmd_Payload Struct Reference	784
11.133.1. Detailed Description	784
11.133.2. Field Documentation	784
11.134. CFE_TIME_StateCmd_Payload Struct Reference	784
11.134.1. Detailed Description	785
11.134.2. Field Documentation	785
11.135. CFE_TIME_SysTime Struct Reference	785
11.135.1. Detailed Description	785
11.135.2. Field Documentation	786
11.136. CFE_TIME_TimeCmd Struct Reference	786
11.136.1. Detailed Description	786
11.136.2. Field Documentation	787
11.137. CFE_TIME_TimeCmd_Payload Struct Reference	787
11.137.1. Detailed Description	787
11.137.2. Field Documentation	787
11.138. CFE_TIME_ToneDataCmd Struct Reference	788
11.138.1. Detailed Description	788
11.138.2. Field Documentation	788
11.139. CFE_TIME_ToneDataCmd_Payload Struct Reference	789
11.139.1. Detailed Description	789
11.139.2. Field Documentation	790
11.140. HS_AMTEEntry_t Struct Reference	791
11.140.1. Detailed Description	791

11.140. Field Documentation	791
11.14 HS_AppData_t Struct Reference	792
11.141. Detailed Description	794
11.141. Field Documentation	794
11.14 HS_CDSData_t Struct Reference	804
11.142. Detailed Description	804
11.142. Field Documentation	804
11.14 HS_CustomData_t Struct Reference	806
11.143. Detailed Description	806
11.143. Field Documentation	806
11.14 HS_EMTEEntry_t Struct Reference	810
11.144. Detailed Description	810
11.144. Field Documentation	811
11.14 HS_HkPacket_t Struct Reference	812
11.145. Detailed Description	812
11.145. Field Documentation	813
11.14 HS_MATEEntry_t Struct Reference	817
11.146. Detailed Description	817
11.146. Field Documentation	817
11.14 HS_MATMMsgBuf_t Union Reference	818
11.147. Detailed Description	818
11.147. Field Documentation	819
11.14 HS_NoArgsCmd_t Struct Reference	819
11.148. Detailed Description	819
11.148. Field Documentation	820
11.14 HS_SetMaxResetsCmd_t Struct Reference	820
11.149. Detailed Description	820
11.149. Field Documentation	820

11.150HS_SetUtilDiagCmd_t Struct Reference	821
11.150.1Detailed Description	822
11.150.2Field Documentation	822
11.151HS_SetUtilParamsCmd_t Struct Reference	822
11.151.1Detailed Description	823
11.151.2Field Documentation	823
11.152HS_XCTEntry_t Struct Reference	824
11.152.1Detailed Description	824
11.152.2Field Documentation	825
11.153S_bin_sem_prop_t Struct Reference	825
11.153.1Detailed Description	826
11.153.2Field Documentation	826
11.154S_count_sem_prop_t Struct Reference	826
11.154.1Detailed Description	827
11.154.2Field Documentation	827
11.155s_dirent_t Struct Reference	827
11.155.1Detailed Description	828
11.155.2Field Documentation	828
11.156S_FdSet Struct Reference	828
11.156.1Detailed Description	828
11.156.2Field Documentation	829
11.157S_file_prop_t Struct Reference	829
11.157.1Detailed Description	829
11.157.2Field Documentation	829
11.158s_fsinfo_t Struct Reference	830
11.158.1Detailed Description	830
11.158.2Field Documentation	830
11.159s_fstat_t Struct Reference	831

11.159.1 Detailed Description	832
11.159.2 Field Documentation	832
11.160 S_heap_prop_t Struct Reference	833
11.160.1 Detailed Description	833
11.160.2 Field Documentation	833
11.161 S_module_address_t Struct Reference	834
11.161.1 Detailed Description	834
11.161.2 Field Documentation	834
11.162 S_module_prop_t Struct Reference	836
11.162.1 Detailed Description	836
11.162.2 Field Documentation	836
11.163 S_mut_sem_prop_t Struct Reference	837
11.163.1 Detailed Description	838
11.163.2 Field Documentation	838
11.164 S_queue_prop_t Struct Reference	838
11.164.1 Detailed Description	838
11.164.2 Field Documentation	839
11.165 S_SockAddr_t Struct Reference	839
11.165.1 Detailed Description	839
11.165.2 Field Documentation	840
11.166 S_SockAddrData_t Union Reference	840
11.166.1 Detailed Description	840
11.166.2 Field Documentation	841
11.167 S_socket_prop_t Struct Reference	841
11.167.1 Detailed Description	842
11.167.2 Field Documentation	842
11.168 S_static_symbol_record_t Struct Reference	842
11.168.1 Detailed Description	843

11.168. Field Documentation	843
11.169. S_statvfs_t Struct Reference	843
11.169.1 Detailed Description	844
11.169.2 Field Documentation	844
11.170. S_task_prop_t Struct Reference	845
11.170.1 Detailed Description	845
11.170.2 Field Documentation	845
11.171. S_time_t Struct Reference	846
11.171.1 Detailed Description	846
11.171.2 Field Documentation	846
11.172. S_timebase_prop_t Struct Reference	847
11.172.1 Detailed Description	847
11.172.2 Field Documentation	847
11.173. S_timer_prop_t Struct Reference	848
11.173.1 Detailed Description	849
11.173.2 Field Documentation	849
12 File Documentation	850
12.1 apps/hs/docs/dox_src/cfs_hs.dox File Reference	850
12.2 apps/hs/fsw/mission_inc/hs_perfids.h File Reference	850
12.2.1 Detailed Description	850
12.3 apps/hs/fsw/platform_inc/hs_msgids.h File Reference	850
12.3.1 Detailed Description	850
12.4 apps/hs/fsw/platform_inc/hs_platform_cfg.h File Reference	851
12.4.1 Detailed Description	852
12.5 apps/hs/fsw/src/hs_app.c File Reference	853
12.5.1 Detailed Description	853
12.5.2 Function Documentation	853

12.5.3 Variable Documentation	864
12.6 apps/hs/fsw/src/hs_app.h File Reference	865
12.6.1 Detailed Description	866
12.6.2 Macro Definition Documentation	866
12.6.3 Function Documentation	867
12.6.4 Variable Documentation	877
12.7 apps/hs/fsw/src/hs_cmds.c File Reference	878
12.7.1 Detailed Description	879
12.7.2 Function Documentation	879
12.8 apps/hs/fsw/src/hs_cmds.h File Reference	898
12.8.1 Detailed Description	899
12.8.2 Function Documentation	899
12.9 apps/hs/fsw/src/hs_custom.c File Reference	918
12.9.1 Detailed Description	919
12.9.2 Function Documentation	919
12.9.3 Variable Documentation	929
12.10 apps/hs/fsw/src/hs_custom.h File Reference	929
12.10.1 Detailed Description	931
12.10.2 Macro Definition Documentation	931
12.10.3 Function Documentation	932
12.10.4 Variable Documentation	941
12.11 apps/hs/fsw/src/hs_events.h File Reference	942
12.11.1 Detailed Description	945
12.12 apps/hs/fsw/src/hs_monitors.c File Reference	945
12.12.1 Detailed Description	946
12.12.2 Function Documentation	946
12.13 apps/hs/fsw/src/hs_monitors.h File Reference	954
12.13.1 Detailed Description	955

12.13.2 Function Documentation	955
12.14apps/hs/fsw/src/hs_msg.h File Reference	964
12.14.1 Detailed Description	964
12.14.2 Macro Definition Documentation	965
12.15apps/hs/fsw/src/hs_msgdefs.h File Reference	965
12.15.1 Detailed Description	966
12.15.2 Macro Definition Documentation	966
12.16apps/hs/fsw/src/hs_tbl.h File Reference	968
12.16.1 Detailed Description	968
12.16.2 Macro Definition Documentation	969
12.17apps/hs/fsw/src/hs_tbldefs.h File Reference	969
12.17.1 Detailed Description	971
12.17.2 Macro Definition Documentation	971
12.18apps/hs/fsw/src/hs_utils.c File Reference	980
12.18.1 Detailed Description	980
12.18.2 Function Documentation	981
12.19apps/hs/fsw/src/hs_utils.h File Reference	984
12.19.1 Detailed Description	984
12.19.2 Function Documentation	984
12.20apps/hs/fsw/src/hs_verify.h File Reference	987
12.20.1 Detailed Description	987
12.21apps/hs/fsw/src/hs_version.h File Reference	987
12.21.1 Detailed Description	987
12.22apps/hs/fsw/tables/hs_amt.c File Reference	988
12.22.1 Detailed Description	988
12.22.2 Function Documentation	988
12.22.3 Variable Documentation	988
12.23apps/hs/fsw/tables/hs_eamt.c File Reference	989

12.23.1 Detailed Description	989
12.23.2 Function Documentation	989
12.23.3 Variable Documentation	989
12.24apps/hs/fsw/tables/hs_mat.c File Reference	990
12.24.1 Detailed Description	990
12.24.2 Function Documentation	990
12.24.3 Variable Documentation	990
12.25apps/hs/fsw/tables/hs_xct.c File Reference	991
12.25.1 Detailed Description	991
12.25.2 Function Documentation	991
12.25.3 Variable Documentation	991
12.26build/docs/osconfig-example.h File Reference	992
12.26.1 Macro Definition Documentation	993
12.27cfe/cmake/sample_defs/cpu1_msgids.h File Reference	1000
12.27.1 Detailed Description	1002
12.27.2 Macro Definition Documentation	1002
12.28cfe/cmake/sample_defs/cpu1_platform_cfg.h File Reference	1009
12.28.1 Detailed Description	1012
12.28.2 Macro Definition Documentation	1012
12.29cfe/cmake/sample_defs/sample_mission_cfg.h File Reference	1066
12.29.1 Detailed Description	1068
12.29.2 Macro Definition Documentation	1068
12.30cfe/cmake/sample_defs/sample_perfids.h File Reference	1087
12.30.1 Detailed Description	1088
12.30.2 Macro Definition Documentation	1088
12.31cfe/docs/src/cfe_api.dox File Reference	1091
12.32cfe/docs/src/cfe_es.dox File Reference	1091
12.33cfe/docs/src/cfe_evs.dox File Reference	1091

12.34cfe/docs/src/cfe_frontpage.dox File Reference	1091
12.35cfe/docs/src/cfe_glossary.dox File Reference	1091
12.36cfe/docs/src/cfe_sb.dox File Reference	1091
12.37cfe/docs/src/cfe_tbl.dox File Reference	1091
12.38cfe/docs/src/cfe_time.dox File Reference	1091
12.39cfe/docs/src/cfe_xref.dox File Reference	1091
12.40cfe/docs/src/cfs_versions.dox File Reference	1091
12.41cfe/modules/core_api/fsw/inc/cfe.h File Reference	1091
12.41.1 Detailed Description	1091
12.42cfe/modules/core_api/fsw/inc/cfe_config.h File Reference	1092
12.42.1 Detailed Description	1092
12.42.2 Function Documentation	1092
12.43cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference	1096
12.43.1 Detailed Description	1097
12.43.2 Macro Definition Documentation	1097
12.43.3 Typedef Documentation	1097
12.44cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference	1098
12.44.1 Detailed Description	1098
12.44.2 Macro Definition Documentation	1098
12.45cfe/modules/core_api/fsw/inc/cfe_error.h File Reference	1098
12.45.1 Detailed Description	1104
12.45.2 Macro Definition Documentation	1105
12.45.3 Typedef Documentation	1107
12.46cfe/modules/core_api/fsw/inc/cfe_es.h File Reference	1107
12.46.1 Detailed Description	1110
12.46.2 Macro Definition Documentation	1110
12.47cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference	1111
12.47.1 Detailed Description	1113

12.47.2 Macro Definition Documentation	1113
12.47.3 Typedef Documentation	1116
12.48cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h File Reference	1118
12.48.1 Detailed Description	1120
12.48.2 Macro Definition Documentation	1120
12.48.3 Typedef Documentation	1121
12.48.4 Enumeration Type Documentation	1127
12.49cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference	1130
12.49.1 Detailed Description	1130
12.49.2 Macro Definition Documentation	1130
12.50cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference	1132
12.50.1 Detailed Description	1132
12.50.2 Macro Definition Documentation	1133
12.50.3 Typedef Documentation	1135
12.51cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h File Reference	1135
12.51.1 Detailed Description	1136
12.51.2 Typedef Documentation	1136
12.51.3 Enumeration Type Documentation	1137
12.52cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference	1140
12.52.1 Detailed Description	1141
12.53cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference	1141
12.53.1 Detailed Description	1141
12.53.2 Typedef Documentation	1142
12.53.3 Enumeration Type Documentation	1143
12.54cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h File Reference	1144
12.54.1 Detailed Description	1145
12.54.2 Macro Definition Documentation	1145
12.54.3 Typedef Documentation	1145

12.54.4 Enumeration Type Documentation	1146
12.55cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference	1147
12.55.1 Detailed Description	1149
12.56cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference	1149
12.56.1 Detailed Description	1151
12.56.2 Macro Definition Documentation	1151
12.56.3 Typedef Documentation	1151
12.56.4 Enumeration Type Documentation	1154
12.57cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference	1157
12.57.1 Detailed Description	1158
12.57.2 Macro Definition Documentation	1158
12.57.3 Function Documentation	1159
12.58cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference	1164
12.58.1 Detailed Description	1165
12.58.2 Macro Definition Documentation	1165
12.59cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference	1166
12.59.1 Detailed Description	1167
12.59.2 Macro Definition Documentation	1167
12.60cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference	1169
12.60.1 Detailed Description	1170
12.60.2 Macro Definition Documentation	1170
12.60.3 Typedef Documentation	1173
12.61cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h File Reference	1173
12.61.1 Detailed Description	1174
12.61.2 Macro Definition Documentation	1174
12.61.3 Typedef Documentation	1175
12.61.4 Enumeration Type Documentation	1176
12.62cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference	1177

12.62.1 Detailed Description	1178
12.63cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference	1178
12.63.1 Detailed Description	1179
12.63.2 Macro Definition Documentation	1180
12.63.3 Typedef Documentation	1180
12.63.4 Enumeration Type Documentation	1181
12.64cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h File Reference	1181
12.64.1 Detailed Description	1182
12.64.2 Typedef Documentation	1182
12.64.3 Enumeration Type Documentation	1182
12.65cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h File Reference	1183
12.65.1 Detailed Description	1183
12.65.2 Macro Definition Documentation	1184
12.65.3 Typedef Documentation	1184
12.66cfe/modules/core_api/fsw/inc/cfe_time.h File Reference	1185
12.66.1 Detailed Description	1186
12.66.2 Macro Definition Documentation	1186
12.67cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference	1187
12.67.1 Detailed Description	1187
12.67.2 Macro Definition Documentation	1187
12.67.3 Typedef Documentation	1188
12.67.4 Enumeration Type Documentation	1188
12.68cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h File Reference	1189
12.68.1 Detailed Description	1190
12.68.2 Typedef Documentation	1190
12.68.3 Enumeration Type Documentation	1192
12.69cfe/modules/core_api/fsw/inc/cfe_version.h File Reference	1195
12.69.1 Detailed Description	1195

12.69.2 Macro Definition Documentation	1196
12.70cfe/modules/es/fsw/inc/cfe_es_events.h File Reference	1198
12.70.1 Detailed Description	1201
12.70.2 Macro Definition Documentation	1201
12.71cfe/modules/es/fsw/inc/cfe_es_msg.h File Reference	1231
12.71.1 Detailed Description	1235
12.71.2 Macro Definition Documentation	1235
12.71.3 Typedef Documentation	1259
12.72cfe/modules/evs/fsw/inc/cfe_evs_events.h File Reference	1267
12.72.1 Detailed Description	1268
12.72.2 Macro Definition Documentation	1268
12.73cfe/modules/evs/fsw/inc/cfe_evs_msg.h File Reference	1281
12.73.1 Detailed Description	1285
12.73.2 Macro Definition Documentation	1285
12.73.3 Typedef Documentation	1306
12.74cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference	1313
12.74.1 Detailed Description	1314
12.74.2 Typedef Documentation	1314
12.75cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference	1314
12.75.1 Detailed Description	1315
12.76cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference	1315
12.76.1 Detailed Description	1316
12.76.2 Macro Definition Documentation	1316
12.77cfe/modules/sb/fsw/inc/cfe_sb_events.h File Reference	1317
12.77.1 Detailed Description	1319
12.77.2 Macro Definition Documentation	1319
12.78cfe/modules/sb/fsw/inc/cfe_sb_msg.h File Reference	1341
12.78.1 Detailed Description	1343

12.78.2 Macro Definition Documentation	1343
12.78.3 Typedef Documentation	1354
12.79cfe/modules/tbl/fsw/inc/cfe_tbl_events.h File Reference	1359
12.79.1 Detailed Description	1361
12.79.2 Macro Definition Documentation	1361
12.80cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h File Reference	1385
12.80.1 Detailed Description	1387
12.80.2 Macro Definition Documentation	1387
12.80.3 Typedef Documentation	1398
12.81cfe/modules/time/fsw/inc/cfe_time_events.h File Reference	1402
12.81.1 Detailed Description	1403
12.81.2 Macro Definition Documentation	1404
12.82cfe/modules/time/fsw/inc/cfe_time_msg.h File Reference	1416
12.82.1 Detailed Description	1419
12.82.2 Macro Definition Documentation	1419
12.82.3 Typedef Documentation	1436
12.83osal/docs/src/osal_frontpage.dox File Reference	1442
12.84osal/docs/src/osal_fs.dox File Reference	1442
12.85osal/docs/src/osal_timer.dox File Reference	1442
12.86osal/src/os/inc/common_types.h File Reference	1442
12.86.1 Detailed Description	1444
12.86.2 Macro Definition Documentation	1444
12.86.3 Typedef Documentation	1445
12.86.4 Function Documentation	1449
12.87osal/src/os/inc/osapi-binsem.h File Reference	1450
12.87.1 Detailed Description	1451
12.88osal/src/os/inc/osapi-bsp.h File Reference	1451
12.88.1 Detailed Description	1452

12.89osal/src/os/inc/osapi-clock.h File Reference	1452
12.89.1 Detailed Description	1453
12.89.2 Enumeration Type Documentation	1453
12.90osal/src/os/inc/osapi-common.h File Reference	1454
12.90.1 Detailed Description	1454
12.90.2 Typedef Documentation	1455
12.90.3 Enumeration Type Documentation	1456
12.91osal/src/os/inc/osapi-constants.h File Reference	1457
12.91.1 Detailed Description	1457
12.91.2 Macro Definition Documentation	1457
12.92osal/src/os/inc/osapi-countsem.h File Reference	1458
12.92.1 Detailed Description	1459
12.93osal/src/os/inc/osapi-dir.h File Reference	1459
12.93.1 Detailed Description	1459
12.93.2 Macro Definition Documentation	1459
12.94osal/src/os/inc/osapi-error.h File Reference	1460
12.94.1 Detailed Description	1462
12.94.2 Macro Definition Documentation	1462
12.94.3 Typedef Documentation	1462
12.95osal/src/os/inc/osapi-file.h File Reference	1463
12.95.1 Detailed Description	1465
12.95.2 Macro Definition Documentation	1465
12.95.3 Enumeration Type Documentation	1466
12.96osal/src/os/inc/osapi-filesystem.h File Reference	1467
12.96.1 Detailed Description	1468
12.96.2 Macro Definition Documentation	1468
12.97osal/src/os/inc/osapi-heap.h File Reference	1469
12.97.1 Detailed Description	1469

12.98osal/src/os/inc/osapi-idmap.h File Reference	1469
12.98.1 Detailed Description	1470
12.98.2 Macro Definition Documentation	1471
12.99osal/src/os/inc/osapi-macros.h File Reference	1471
12.99.1 Detailed Description	1471
12.99.2 Macro Definition Documentation	1472
12.100sal/src/os/inc/osapi-module.h File Reference	1473
12.100.1 Detailed Description	1474
12.100.2 Macro Definition Documentation	1474
12.101sal/src/os/inc/osapi-mutex.h File Reference	1475
12.101.1 Detailed Description	1476
12.102sal/src/os/inc/osapi-network.h File Reference	1476
12.102.1 Detailed Description	1476
12.103sal/src/os/inc/osapi-printf.h File Reference	1476
12.103.1 Detailed Description	1476
12.104sal/src/os/inc/osapi-queue.h File Reference	1477
12.104.1 Detailed Description	1477
12.105sal/src/os/inc/osapi-select.h File Reference	1477
12.105.1 Detailed Description	1478
12.105.2 Enumeration Type Documentation	1478
12.106sal/src/os/inc/osapi-shell.h File Reference	1479
12.106.1 Detailed Description	1479
12.107sal/src/os/inc/osapi-sockets.h File Reference	1479
12.107.1 Detailed Description	1481
12.107.2 Macro Definition Documentation	1481
12.107.3 Enumeration Type Documentation	1481
12.108sal/src/os/inc/osapi-task.h File Reference	1482
12.108.1 Detailed Description	1483

12.108.1Macro Definition Documentation	1483
12.108.2Typedef Documentation	1484
12.108.3Function Documentation	1485
12.109sal/src/os/inc/osapi-timebase.h File Reference	1485
12.109.1Detailed Description	1486
12.109.2Typedef Documentation	1486
12.110sal/src/os/inc/osapi-timer.h File Reference	1486
12.110.1Detailed Description	1487
12.110.2Typedef Documentation	1487
12.111sal/src/os/inc/osapi-version.h File Reference	1487
12.111.1Detailed Description	1488
12.111.2Macro Definition Documentation	1488
12.111.3Function Documentation	1491
12.112sal/src/os/inc/osapi.h File Reference	1493
12.112.1Detailed Description	1493
12.113psp/fsw/inc/cfe_psp.h File Reference	1493
12.113.1Macro Definition Documentation	1496
12.113.2Function Documentation	1505

1 CFS Health and Safety (HS) Documentation

- [CFS Health and Safety Introduction](#)
- [CFS Health and Safety Overview](#)
- [CFS Health and Safety Operation](#)
- [CFS Health and Safety Commands](#)
- [CFS Health and Safety Telemetry](#)
- [CFS Health and Safety Events](#)
- [CFS Health and Safety Deployment Guide](#)
- [CFS Health and Safety Configuration](#)
- [CFS Health and Safety Table Definitions](#)
- [CFS Health and Safety Operational Constraints](#)
- [CFS Health and Safety Frequently Asked Questions](#)

1.1 CFS Health and Safety Introduction

Scope

This document provides a complete specification for the commands and telemetry associated with the Health and Safety (HS) application software. The document is intended primarily for users of the software (operations personal, test engineers, and maintenance personnel). The last section of the document, the deployment guide section, is intended for mission developers when deploying and configuring the HS application software for a mission flight software build environment.

CFS Health and Safety 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
HS	Health and Safety
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
TDM	Time Data Multiplex
TLM	Telemetry
UTC	Universal time code

1.2 CFS Health and Safety Overview

The CFS Health and Safety (HS) application provides functionality for Application Monitoring, Event Monitoring, Hardware Watchdog Servicing, Execution Counter Reporting (optional), and CPU Aliveness Indication (via UART).

Application Monitoring, Event Monitoring, and Execution Counter Reporting are configurable by table while the application is running. Application Monitoring, Event Monitoring and CPU Aliveness Indication can be disabled or enabled by ground command, and can be configured to be disabled or enabled (if a table is provided) on startup.

1.2.1 Design Overview

The Health and Safety application nominally runs once per second. First, it makes updates based on any table loads during the previous second. Next, it performs Application Monitoring, if enabled, in the order listed in the table, taking actions if necessary. Next, the Aliveness Indicator (to the UART) is output if enabled and it is time to do so. After that, all events received during the previous cycle are checked, in order, if enabled, against the Event Monitor Table (in the order listed in the table) and actions are taken if necessary. Next, all commands and housekeeping requests received during the previous cycle are processed. Finally, the hardware watchdog is serviced after which the HS application waits until the next cycle to begin the process again.

1.3 CFS Health and Safety Operation

The Health and Safety Application consists of several mostly independent components. The following describes these components and their potential uses.

Application Monitor

The HS Application Application Monitor (AppMon), is used to take action when a table defined application fails to increment its Execution Counter for the table defined number of cycles. The HS Application will not start monitoring applications until system startup has been completed ([CFE_ES_WaitForStartupSync](#) returns, whether because the system is finished starting up, or because it timed out).

AppMon checks the execution count each cycle for each application defined in the Application Monitor Table. If the current value of the execution counter matches the value of the counter during the previous cycle then the "missing" count is decremented, otherwise the "missing" count is reset to the table defined threshold count value. If the "missing" count reaches zero, then the table defined action is taken.

Once the "missing" count reaches zero and the action is taken, that table entry is disabled until Application Monitoring (as a whole) is commanded to be enabled (it does not matter if Application Monitoring is disabled first), or a new Application Monitor Table is loaded.

An application may appear in the Application Monitor Table more than once, allowing it to have multiple actions. This might be used to attempt to restart an application, and failing that (having a larger timeout threshold) perform a processor reset. Another use might be to take another action (perhaps a Message Action causing a Power-On Reset) if the HS Max Processor Reset limit has been reached. This ability also might be used for sending multiple Message Actions.

If an application is not currently running on the system, then it is considered to not be incrementing its counter (which does not exist). The HS Application itself does not know whether the Application is legitimately missing, or was incorrectly specified in the table, so it assumes that it is a real, missing application. If Processor Reset Limiting is not set properly, such an application missing at startup could even lead to an infinite reset loop.

Event Monitor

The HS Application Event Monitor (EventMon) is used to take action when a table defined application generates an event with a table defined Event ID number. The HS Application will not start monitoring events until system startup has been completed ([CFE_ES_WaitForStartupSync](#) returns, whether because the system is finished starting up, or because it timed out). Note that EventMon does not monitor the contents of events, only the generation of them (by Event ID number).

EventMon can only monitor events that have not been filtered by Event Services (EVS). If EVS filters out an event, it will not be sent out, and so HS will never receive it.

EventMon checks all generated events once each cycle, checking the events generated during the previous cycle. For each event received, the Event ID number is checked against the each Event ID number in the Event Monitor Table; if the Event ID number matches, then the Application Name is compared, and if it too matches, then the event monitor takes the table specified action. If the Event Monitor Table contains multiple instances of an Application Name/Event ID number pair, then multiple actions will be taken in the order listed in the table. If one of the multiple actions is a Processor Reset action, and the HS Application has not reached its maximum number of Processor Reset attempts (and so a reset occurs), then no further actions would be taken.

Message Actions

Message Actions allow the HS Application to send a message on the software bus by an action type from the Application Monitor or Event Monitor. While this would typically be used to send a command, it is also possible to send a telemetry message. Each Message Action only sends a single message, but Application Monitoring or Event Monitoring tables can be set up to perform multiple actions for the same application or event if multiple messages are needed. Each Message Action has its own action type.

The Message Actions table can specify a cooldown for each message it can send; the cooldown determines how many cycles must be waited before the message can be sent again. For example, if a message has a cooldown value of 4, and is sent due to a monitored event, then if the same event is received again in the next three cycles, no message would be sent; if it is received again on the 4th cycle or later, the message will be sent again. A cooldown value of 1 means the message can be sent once per cycle. A cooldown value of 0 means the message can be sent multiple times per cycle.

Processor Reset Limiting

The HS Application limits the number of Processor Resets that it will perform to prevent the system from getting into an infinite reset loop. HS keeps track of how many processor resets it has performed in a Critical Data Store; if this CDS is corrupt or does not exist (due to design or Power-On reset) then it assumes 0 resets have been performed by HS. Because Executive Services (ES) will perform a Power-On reset after a defined threshold of Processor Resets have occurred, the threshold value for HS Processor Resets should be set to less than this, if the developer wants to use HS Processor Reset Limiting (conversely, it should be set higher if the developer does not want to use limiting). A Max HS Processor Reset value of 0 means that HS will never attempt to perform a processor reset.

If there is the desire to bypass Reset Limiting for some but not all situations (especially if a Power-On reset might be necessary), then Message Actions can be used to command a reset via ES. HS will not consider this an HS caused Processor Reset and will not increment its counter for this.

Watchdog Management

The HS Application enables and initializes the timeout value of the hardware watchdog once at startup. Note that this is the only time that the HS Application will set the watchdog timeout via the [CFE_PSP_WatchdogSet\(\)](#) function.

The HS Application nominally services the hardware watchdog once every cycle using [CFE_PSP_WatchdogService\(\)](#). The HS Application will only disable this periodic servicing of the watchdog when it attempts to perform a processor reset (in response to Event or Application monitoring) as a contingency against [CFE_ES_ResetCFE\(\)](#) failing; nominally the processor reset will re-initialize HS which will once again initialize and service the watchdog.

If the developer wishes to occasionally set a longer watchdog timeout in situations where HS may be prevented from running by a higher priority task for extended periods of time, this may be accomplished using the [CFE_PSP_WatchdogSet\(\)](#) API with a larger timeout value. As the HS Application only sets the timeout value at startup, if it does run during that time the watchdog will be reset to larger value. After setting the new timeout value, the developer should service using the watchdog using [CFE_PSP_WatchdogService\(\)](#) to ensure that the new timeout value is being used. Prior to setting the new timeout value, the developer should get the current timeout value using [CFE_PSP_WatchdogGet\(\)](#), and restore this value after the larger value is no longer needed.

Execution Counter Reporting

The HS Application provides the ability to report execution counter data in telemetry. Execution counter data can be reported for both applications and child tasks.

If the application or task specified in the a table entry fails to be found (either due to absence or improper naming), or if no application or task is specified, the telemetry associated with that entry will read 0xFFFFFFFF. If no table is present, then all associated telemetry will read 0xFFFFFFFF.

The Execution Counter Telemetry reporting functionality is optional, and is not included in the build process if no counters will be reported (the Execution Counter Table to support this functionality would also not exist in such a case).

Aliveness Indication

The Aliveness indicator will, if enabled (either by command or by configuration parameter), output the predefined configuration parameter string using [OS_printf](#) at the configuration parameter determined rate.

CPU Utilization Monitoring and Hogging Detection

The HS Application creates an Idle Child Task at a low priority, that is used to determine the portion of CPU utilization not being used by other applications. The Idle Child Task continually increments a counter while running; and this counter is recorded at 1 Hz by a TIME 1 Hz callback function. When the HS application runs each cycle, it will compute the non-Idle Child Task utilization that occurred during the previous cycle and report the average utilization, peak utilization, and determine if the processor is being hogged. The frequency at which the count is recorded and at which the HS application computes utilization can be configured to be other than 1 Hz or 1 cycle if necessary.

If the utilization is in excess of a configurable amount for a configurable period of time, then the CPU Hogging event message will be sent. This message can be used to by HS Event Monitoring to perform whatever action the user desires.

The Idle Child Task requires calibration to perform properly, and provides the ability to perform the necessary calibrations using the software itself. See the deployment portion of this guide for more information on calibration.

1.4 CFS Health and Safety Commands

[CFS Health and Safety Command Message IDs](#)

[CFS Health and Safety Command Structures](#)

[CFS Health and Safety Command Codes](#)

1.5 CFS Health and Safety Telemetry

[CFS Health and Safety Telemetry Message IDs](#)

[CFS Health and Safety Telemetry](#)

1.6 CFS Health and Safety Events

[CFS Health and Safety Event IDs](#)

1.7 CFS Health and Safety Deployment Guide

The Health and Safety application cycle may be any length of time, but typically 1 Hz is chosen as the desired rate. The cycle time is important when considering all of the configuration parameters and directly affects the values that should be chosen for Application Monitoring. The cycle is initiated by pending on a Software Bus pipe for a packet with the [HS_WAKEUP_MID](#) Message ID. This pend may be forever, or may be set to a specific timeout value using the [HS_WAKEUP_TIMEOUT](#) configuration parameter to make sure HS runs even if no message is received. This configuration parameter can even be used to make HS run independently if no wakeup message is expected. Additionally, there is a configuration parameter called [HS_POST_PROCESSING_DELAY](#) which will cause a task delay for the configured number of milliseconds after the HS processing cycle is over, before it pends on the Software Bus for the next packet. Just as a general note, configuration of how HS should run should be carefully considered in the context of the system it will be run on and the desires of the project. As some examples:

On a system with a fairly reliable schedule, the easiest method for setting up HS to run could be to add the [HS_WAKEUP_MID](#) to the Scheduler (SCH) Application to be sent once per second, setting [HS_POST_PROCESSING_DELAY](#) to 0 and [HS_WAKEUP_TIMEOUT](#) to [CFE_SB_PEND_FOREVER](#) (or to greater than a second if assurance that HS runs is desired).

On a subsystem where scheduling may be reset by an outside source or delayed by a resource intensive task, this design may not be as desirable as it could interfere with Application Monitoring. In such a case, either an alternate, more reliable, source could be used for 1 Hz delivery, or [HS_POST_PROCESSING_DELAY](#) can be used to assure that the cycle time is not too short in cases of a reset.

If more independence of HS is desired, [HS_POST_PROCESSING_DELAY](#) could be used for scheduling with [HS_WAKEUP_TIMEOUT](#) set to [CFE_SB_POLL](#) (though this method would not have high timing accuracy).

Each of the Health and Safety Application tables is sized by a configuration parameter. It is probably a good idea to leave a reasonable number of spare entries. Remember that applications can appear in the Application Monitor Table more than once, and events can appear in the Event Monitor Table more than once as well, making complex actions possible (provided there are enough table entries to use).

Special consideration needs to be given to several configuration parameters:

- [HS_EVENT_PIPE_DEPTH](#) - This parameter determines how many events can be stored for Event Monitoring each cycle, and so should be deep enough to capture the maximum number of expected events.
- [HS_STARTUP_SYNC_TIMEOUT](#) - This parameter needs to be set so that HS will start after all applications being monitored.
- [HS_MAX_RESTART_ACTIONS](#) - This parameter needs to be set with consideration to [CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS](#) as if Executive Services performs a Power-On Reset, the count of resets stored by HS will be reset. (This may be desirable if HS reset limiting is not desired)

Health and Safety CPU Utilization Monitoring Customization

The Health and Safety application contains the file [hs_custom.c](#) which can be adapted to take advantage of platform specific functionality concerning CPU Utilization Monitoring. The delivered implementation contains a generic version which should work well on a variety of platforms, however it may be possible to obtain better accuracy or increase processing efficiency by modifying the contents. As the custom utilization monitoring function is called every HS cycle, and an interface to add additional commands is included, it could even be used as a way to add other capabilities to the HS application (e.g. offboard watchdog servicing, or floating point exception testing) instead of creating an entirely separate application.

This customizable section of code includes five required functions for interface purposes:

- [HS_CustomInit](#)
- [HS_CustomCommands](#)
- [HS_CustomMonitorUtilization](#)
- [HS_CustomGetUtil](#)
- [HS_CustomCleanup](#)

The remainder of this guide assumes that the generic [hs_custom.c](#) is being used when describing how to use and calibrate CPU Utilization Monitoring.

Health and Safety CPU Utilization Monitoring Calibration

The first step in CPU Utilization Monitoring Calibration is to determine how many Idle Child Task cycles there are in a second (in the default case: one interval is one second). This is done by setting the Utilization Diagnostics Mask by command: for example a command parameter of 0xFF would record the time every 256 cycles. These times are stored in a circular buffer of a size determined by [HS_UTIL_TIME_DIAG_ARRAY_POWER](#) (increasing this parameter will provide more samples, but use more memory). The Report Diagnostics command can then be sent (after a sufficient time to let the circular buffer perform a full cycle) to generate an event message containing the mask, and the four shortest intervals between time records, as well as how frequently those timed intervals occurred. Usually the desired value is the weighted average (based on frequencies) of the two lowest times, though common sense should be used as a sanity check. The Report Diagnostics command can be used multiple times to report additional captured times (the capturing continually runs through the circular buffer). The recorded time uses the microseconds field of [CFE_PSP_GetTime](#) to determine the time, so the source of that timer will determine the precision and accuracy of the timing data. To calibrate the time properly, two values must be used so that the overhead associated with recording the time is cancelled out (so for example, capturing time at 256 and 128 cycles, and then finding the difference will give the real time for performing 128 cycles). When done collecting data, reset the mask to 0xFFFFFFFF. This makes sure that no timing data is captured, and so all cycles are of equal length.

Good rates at which to record depend on the timing resolution and platform speed. Time is recorded when the mask value is all 1s, so the mask values used should be of the form $2^n - 1$ (e.g. 255 (0xFF), 2047 (0x7FF)) to capture once out of 2^n cycles. As the calibration is occurring while the system is running, it is necessary to have large enough chunks of idle time to capture several times within a chunk; with high resolution timers on a fast platform, collecting good data may be possible even when the processor is mostly utilized, but with lower time resolution special testing may have to be done with other applications absent. Reported times of 0 imply that the mask is much too small relative to the resolution (though if all times are 0, the mask may be too large such that no times have been recorded, or that [CFE_PSP_GetTime](#) was not configured properly for the platform). If the frequencies of all the lowest times are relatively low, it may imply that the mask is too large relative to the idle time blocks, or that [CFE_PSP_GetTime](#) is not running in constant time and the resolution is high enough to be affected by that (in which case either the lowest, or average of several lowest times can be used). Once the timing information has been acquired, it can be used to determine the time per Idle Child Task cycle, and from there, the number of Idle Child Task cycles per second.

On some platforms calibration can also be done using a logic analyzer with the performance monitor using ID [HS_IDLETASK_PERF_ID](#), potentially supplying much higher time resolution. In addition this marker can also be used by dumping performance data with different masks set which may (or may not) provide higher resolution timing data as the performance monitor uses a different timer.

The second step is to choose the [HS_UTIL_PER_INTERVAL_TOTAL](#) value: this is equivalent to one full interval of CPU utilization, so if a value of 100 was used, then the value reported in telemetry would be the percentage. Alternatively a higher value (10000 or 100000) could provide higher resolution information if the calibration is of high enough precision and accuracy. The calibration is then configured using the following equation:

$$\text{HS_UTIL_PER_INTERVAL_TOTAL} = (((\text{Max Number of Idle Task Cycles per Second} * \text{HS_UTIL_CONV_MULT1}) / \text{HS_UTIL_CONV_DIV}) * \text{HS_UTIL_CONV_MULT2})$$

The multiplication and division parameters should be set so as to prevent overflows (of 32 bit values) and using any other platform considerations. These values can be set by command, but they also exist as configuration parameters so that if the values are known at compile time for a specific platform, no further calibration would be necessary.

Also, please be aware that HS has a performance ID, [HS_APPMAIN_PERF_ID](#), that keeps track of the performance of the HS app.

1.8 CFS Health and Safety Configuration

[CFS Health and Safety Mission Configuration](#)

[CFS Health and Safety Platform Configuration](#)

1.9 CFS Health and Safety Table Definitions

The CFS Health and Safety application utilizes three (if not using Execution Counter Reporting) or four tables.

Application Monitor Table (AMT)

The AMT is the configuration table for the HS Application Monitor, defining when to take action and what action to take.

The AMT consists of an array of records numbering [HS_MAX_MONITORED_APPS](#) defined by [HS_AMTEntry_t](#).

- The Application Name should be the name of the application as the system knows it: up to [OS_MAX_API_NAME](#) in length.
- The Null Terminator must be 0: this exists as quick method making sure that the strings in the table are no longer than the maximum length.
- The Cycle Count is the number of HS cycles that the named application's execution counter must not be incremented before the defined action is taken.
- The Action Type is the action to take:
 - [HS_AMT_ACT_NOACT](#) for a disabled entry.
 - [HS_AMT_ACT_PROC_RESET](#) for an entry that causes a processor reset.
 - [HS_AMT_ACT_APP_RESTART](#) for an entry that restarts the named application.
 - [HS_AMT_ACT_EVENT](#) for an entry that only generates an event message.
 - [HS_AMT_ACT_MSG\(num\)](#) for an entry that generates a message action, where 'num' is the index into the Message Actions Table.

While there is no valid AMT loaded, the HS Application Monitor will be disabled (it will be disabled again if an attempt to enable it is made).

Event Monitor Table (EMT)

The EMT is the configuration table for the HS Event Monitor, defining when to take action and what action to take.

The EMT consists of an array of records numbering [HS_MAX_MONITORED_EVENTS](#) defined by [HS_EMTEEntry_t](#).

- The Application Name should be the name of the application as the system knows it: up to [OS_MAX_API_NAME](#) in length.
- The Null Terminator must be 0: this exists as quick method making sure that the strings in the table are no longer than the maximum length.
- The Event ID is the number of the event being monitored for.
- The Action Type is the action to take:
 - [HS_EMT_ACT_NOACT](#) for a disabled entry.
 - [HS_EMT_ACT_PROC_RESET](#) for an entry that causes a processor reset.
 - [HS_EMT_ACT_APP_RESTART](#) for an entry that restarts the named application.
 - [HS_EMT_ACT_APP_DELETE](#) for an entry that deletes the named application.
 - [HS_EMT_ACT_MSG\(num\)](#) for an entry that generates a message action, where 'num' is the index into the Message Actions Table.

While there is no valid EMT loaded, the HS Event Monitor will be disabled (it will be disabled again if an attempt to enable it is made).

Message Actions Table (MAT)

The MAT stores the messages sent by the HS Application Monitor or HS Event Monitor Message Action Type.

The MAT consists of an array of records numbering [HS_MAX_MSG_ACT_TYPES](#) defined by [HS_MATEEntry_t](#).

- The Cooldown value determines how many cycles must be waited before a message can be sent again; a value of 0 means the message can be sent multiple times in one cycle, a value of 1 means the message can only be sent once per cycle, a value of 2 means the message could only be sent every other cycle, etc.
- The Message is an array of bytes of length [HS_MAX_MSG_ACT_SIZE](#) which contains the message to be sent. Keep byte-swapping issues in mind when populating this field.
- The Enable State determines whether the message action in that record can be sent:
 - [HS_MAT_STATE_DISABLED](#) for a disabled entry.
 - [HS_MAT_STATE_ENABLED](#) for an enabled entry which will generate an event when sent.
 - [HS_MAT_STATE_NOEVENT](#) for an enabled entry which will not generate an event when sent.

While there is no valid MAT loaded, no Message Actions will be sent.

Execution Counter Table (XCT)

The EMT is the configuration table for the HS Execution Counter reporting, defining which counters are reported in telemetry.

The XCT consists of an array of records numbering [HS_MAX_EXEC_CNT_SLOTS](#) defined by [HS_XCTEntry_t](#).

- The Resource Name should be the name of the application or child task as the system knows it: up to [OS_MAX_API_NAME](#) in length.
- The Null Terminator must be 0: this exists as quick method making sure that the strings in the table are no longer than the maximum length.
- The Resource Type is the type of resource being monitored:
 - [HS_XCT_TYPE_NOTYPE](#) for a disabled entry.
 - [HS_XCT_TYPE_APP_MAIN](#) for an Application Main Task.
 - [HS_XCT_TYPE_APP_CHILD](#) for a Child Task.
 - [HS_XCT_TYPE_ISR](#) for an ISR counter.
 - [HS_XCT_TYPE_DEVICE](#) for a Device Driver counter.

While there is no valid XCT loaded, all Execution Counters in telemetry will be reported as invalid (0xFFFFFFFF).

1.10 CFS Health and Safety Operational Constraints

The Health and Safety application is very robust and will only fail to run or stop running if certain cFE failures occur. At start-up, this includes registering tables with Table Services, subscribing to Event Services, and creating pipes and subscribing with Software Bus. While running, this only occurs if the Software Bus Receive API generates an error.

While the typical operational scenario for changing the Application Monitor Table or Event Monitor Table would probably involve disabling the respective Monitoring type, the software does support loading while running (although this will reset the Application Monitoring even if the same or a similar table is loaded).

1.11 CFS Health and Safety Frequently Asked Questions

(Q) Why is there no option to start an RTS in response to Application Monitor failure or Event Monitor detection?

RTSes may still be started using Message Actions, provided your system is using the Stored Command (SC) Application. Message Actions were added in favor of retaining RTS capabilities as they provide a more generic solution (such as systems without SC or with an alternative type of SC).

(Q) What if no events need to be monitored?

The Event Monitor Table must exist (and so must be of non-zero length), but Event Monitoring can be disabled by default. Event Monitoring may always be useful at some point in the future life of a mission, so be sure that enough spare entries are available (even if all entries are spares).

(Q) What if no Message Actions are needed?

The Message Actions Table must exist (and so must be of non-zero length), but all entries in it can be disabled. Message Actions may always be useful at some point in the future life of a mission, so be sure that enough spare entries are available (even if all entries are spares).

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.

Core Flight Executive (cFE) Goals

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

Functional and Community Goals

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

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

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

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.

How and Where Defined

The version numbers are provided as simple macros defined in the [cfi_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).

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.

Templates for the short and long version string

See [cfi_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_V←RSION" (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
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

Acronym	Description
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](#)
 - [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
- Executive Services Frequently Asked Questions

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"

Next: "[Application](#)" and "[cFE Application](#)"
Up To: [cFE Executive Services Overview](#)

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.

Next: "[Task](#)"

Up To: [Terminology](#)

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.

Next: "[Startup Script](#)"

Prev: "[Application](#)" and "[cFE Application](#)"

Up To: [Terminology](#)

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.

Next: [Software Reset](#)

Prev: [Starting an Application](#)

Up To: [Terminology](#)

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](#).

Next: [Reset Types and Subtypes](#)

Prev: [Terminology](#)

Up To: [cFE Executive Services Overview](#)

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_SPEICIAL_COMMAND](#), [CFE_PSP_RST_SUBTYPE_HW_WATCHDOG](#), [CFE_PSP_RST_SUBTYPE_RESET_COMMAND](#), [CFE_PSP_RST_SUBTYPE_EXCEPTION](#), [CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET](#), [CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET](#), [CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET](#).

Next: [Exception and Reset \(ER\) Log](#)

Prev: [Software Reset](#)

Up To: [cFE Executive Services Overview](#)

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](#).

Next: [Application and Child Task Management](#)

Prev: [Reset Types and Subtypes](#)

Up To: [cFE Executive Services Overview](#)

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.

Next: [Starting an Application](#)

Prev: [Software Reset](#)

Up To: [cFE Executive Services Overview](#)

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.

Next: [Stopping an Application](#)

Up To: [Application and Child Task Management](#)

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_AppNameCmd_t](#). The only parameter in the command is an application name.

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

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

Next: [Restarting an Application](#)

Prev: [Starting an Application](#)

Up To: [Application and Child Task Management](#)

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.

Next: [Reloading an Application](#)

Prev: [Stopping an Application](#)

Up To: [Application and Child Task Management](#)

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.

Next: [Listing Current Applications](#)

Prev: [Restarting an Application](#)

Up To: [Application and Child Task Management](#)

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](#).

Next: [Listing Current Tasks](#)

Prev: [Reloading an Application](#)

Up To: [Application and Child Task Management](#)

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

Next: [Loading Common Libraries](#)

Prev: [Listing Current Applications](#)

Up To: [Application and Child Task Management](#)

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.

Next: [Basic File System](#)

Prev: [Listing Current Tasks](#)Up To: [Application and Child Task Management](#)

2.6.13 Basic File System

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

Next: [Performance Data Collection](#)

Prev: [Loading Common Libraries](#)Up To: [Application and Child Task Management](#)

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](#)

Next: [Performance Data Collection Trigger Masks](#)

Prev: [Basic File System](#)Up To: [cFE Executive Services Overview](#)

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.

Next: [Starting to Collect Performance Data](#)

Prev: [Performance Data Collection](#)

Up To: [Performance Data Collection](#)

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

Next: [Stopping the Collection of Performance Data](#)

Prev: [Performance Data Collection Trigger Masks](#)

Up To: [Performance Data Collection](#)

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.

Next: [Viewing the Collection of Performance Data](#)

Prev: [Starting to Collect Performance Data](#)

Up To: [Performance Data Collection](#)

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.

Next: [Critical Data Store](#)

Prev: [Stopping the Collection of Performance Data](#)

Up To: [Performance Data Collection](#)

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](#)).

Next: [Memory Pool](#)

Prev: [Performance Data Collection](#)

Up To: [cFE Executive Services Overview](#)

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

Next: [System Log](#)

Prev: [Critical Data Store](#)

Up To: [cFE Executive Services Overview](#)

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_SYSLOG_CC](#) command must be sent. This command will write the log to a binary file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter [CFE_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_SYSLOG_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.

Next: [Version Identification](#)

Prev: [Memory Pool](#)

Up To: [cFE Executive Services Overview](#)

2.6.18 Version Identification

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

Next: [Executive Services Frequently Asked Questions](#)

Prev: [System Log](#)

Up To: [cFE Executive Services Overview](#)

2.6.19 Executive Services Frequently Asked Questions

Prev: [Version Identification](#)

Up To: [cFE Executive Services Overview](#)

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_SYSLOG_CC](#)

Clear Executive Services System Log

Global CFE_ES_DELETE_CDS_CC

Delete Critical Data Store

Global CFE_ES_DUMP_CDS_REGISTRY_CC

Dump Critical Data Store Registry to a File

Global CFE_ES_NOOP_CC

Executive Services No-Op

Global CFE_ES_OVER_WRITE_SYSLOG_CC

Set Executive Services System Log Mode to Discard/Overwrite

Global CFE_ES_QUERY_ALL_CC

Writes all Executive Services Information on all 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_SYSLOG_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_OneAppTlm_Payload_t

Single Application Information Packet

Global CFE_ES_OneAppTlm_Payload_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

Global CFE_MISSION_ES_CDS_MAX_NAME_LENGTH

Maximum Length of CDS Name

Global CFE_MISSION_ES_DEFAULT_CRC

Mission Default CRC algorithm

Global CFE_MISSION_ES_MAX_APPLICATIONS

Mission Max Apps in a message

Global CFE_MISSION_ES_PERF_MAX_IDS

Define Max Number of Performance IDs for messages

Global CFE_MISSION_ES_POOL_MAX_BUCKETS

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

Global CFE_PLATFORM_ES_APP_SCAN_RATE

Define ES Application Control Scan Rate

Global CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES

Define Maximum Number of Registered CDS Blocks

Global CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01

Define ES Critical Data Store Memory Pool Block Sizes

Global CFE_PLATFORM_ES_CDS_SIZE

Define Critical Data Store Size

Global CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE

Default Application Information Filename

Global CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE

Default Critical Data Store Registry Filename

Global CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE

Default Exception and Reset (ER) Log Filename

Global CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME

Default Performance Data Filename

Global CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE

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

Global CFE_PLATFORM_ES_DEFAULT_STACK_SIZE

Define Default Stack Size for an Application

Global CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE

Default System Log Filename

Global CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE

Default Application Information Filename

Global CFE_PLATFORM_ES_ER_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

Global CFE_PLATFORM_ES_MAX_APPLICATIONS

Define Max Number of Applications

Global CFE_PLATFORM_ES_MAX_GEN_COUNTERS

Define Max Number of Generic Counters

Global CFE_PLATFORM_ES_MAX_LIBRARIES

Define Max Number of Shared libraries

Global CFE_PLATFORM_ES_MAX_MEMORY_POOLS

Maximum number of memory pools

Global CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS

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

Global CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN

Define Memory Pool Alignment Size

Global CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING

Default virtual path for persistent storage

Global CFE_PLATFORM_ES_NONVOL_STARTUP_FILE

ES Nonvolatile Startup Filename

Global CFE_PLATFORM_ES_OBJECT_TABLE_SIZE

Define Number of entries in the ES Object table

Global CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY

Define Performance Analyzer Child Task Delay

Global CFE_PLATFORM_ES_PERF_CHILD_PRIORITY

Define Performance Analyzer Child Task Priority

Global CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE

Define Performance Analyzer Child Task Stack Size

Global CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE

Define Max Size of Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS

Define Performance Analyzer Child Task Number of Entries Between Delay

Global CFE_PLATFORM_ES_PERF_FILTMASK_ALL

Define Filter Mask Setting for Enabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_FILTMASK_INIT

Define Default Filter Mask Setting for Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_FILTMASK_NONE

Define Filter Mask Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_TRIGMASK_ALL

Define Filter Trigger Setting for Enabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_TRIGMASK_INIT

Define Default Filter Trigger Setting for Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_TRIGMASK_NONE

Define Default Filter Trigger Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_POOL_MAX_BUCKETS

Maximum number of block sizes in pool structures

Global CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING

Default virtual path for volatile storage

Global CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS

ES Ram Disk Number of Sectors

Global CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED

Percentage of Ram Disk Reserved for Decompressing Apps

Global CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE

ES Ram Disk Sector Size

Global CFE_PLATFORM_ES_RESET_AREA_SIZE

Define ES Reset Area Size

Global CFE_PLATFORM_ES_START_TASK_PRIORITY

Define ES Task Priority

Global CFE_PLATFORM_ES_START_TASK_STACK_SIZE

Define ES Task Stack Size

Global CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC

Startup script timeout

Global CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC

Poll timer for startup sync delay

Global CFE_PLATFORM_ES_SYSTEM_LOG_SIZE

Define Size of the cFE System Log.

Global CFE_PLATFORM_ES_USER_RESERVED_SIZE

Define User Reserved Memory Size

Global CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE

ES Volatile Startup Filename

Global CFE_PLATFORM_EVS_START_TASK_PRIORITY

Define EVS Task Priority

Global CFE_PLATFORM_EVS_START_TASK_STACK_SIZE

Define EVS Task Stack Size

Global CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01

Define SB Memory Pool Block Sizes

Global CFE_PLATFORM_SB_START_TASK_PRIORITY

Define SB Task Priority

Global CFE_PLATFORM_SB_START_TASK_STACK_SIZE

Define SB Task Stack Size

Global CFE_PLATFORM_TBL_START_TASK_PRIORITY

Define TBL Task Priority

Global CFE_PLATFORM_TBL_START_TASK_STACK_SIZE

Define TBL Task Stack Size

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
- 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.

Next: [Local Event Log](#)

Up To: [cFE Event Services Overview](#)

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](#).

Local Event Log Mode

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

Next: [Event Message Control](#)

Prev: [Event Message Format](#)

Up To: [cFE Event Services Overview](#)

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.

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.

Event Message Control - By Application

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

Event Message Control - By Event Type for an Application

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

Event Message Control - Individual Events

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

Modifying a registered event message filter

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

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

Adding/Removing an event message for filtering

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

An on-orbit mission, for example, might be experiencing a problem resulting in 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.

Next: [Event Message Filtering](#)

Prev: [Local Event Log](#)

Up To: [cFE Event Services Overview](#)

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.

Next: [EVS Registry](#)

Prev: [Event Message Control](#)

Up To: [cFE Event Services Overview](#)

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

Next: [EVS Counters](#)

Prev: [Event Message Filtering](#)

Up To: [cFE Event Services Overview](#)

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.

Next: [Resetting EVS Counters](#)

Prev: [EVS Registry](#)

Up To: [cFE Event Services Overview](#)

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.

Next: [Effects of a Processor Reset on EVS](#)

Prev: [EVS Counters](#)

Up To: [cFE Event Services Overview](#)

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.

Next: [Frequently Asked Questions about Event Services](#)

Prev: [Resetting EVS Counters](#)

Up To: [cFE Event Services Overview](#)

2.10.9 Frequently Asked Questions about Event Services

(Q) My telemetry stream is being flooded with the same event message. How do I make it stop?	
	The most direct way to stop an event message from flooding your downlink stream is to send a command to EVS to filter the offending event (see Event Message Control or \$sc_\$cpu_EVS_SetBinFltrMask). In order to stop the event message from being sent, a bit mask of '0xFFFF' should be used. If the event is not currently registered for filtering, the event message must be added using the command \$sc_\$cpu_EVS_AddEvtFltr .
(Q) I filtered an event message and would now like to see it again. What do I do in order to see those events again?	
	If the event message that you are interested in is registered with EVS for filtering, then you have 2 options:
<ol style="list-style-type: none"> 1. You can use the \$sc_\$cpu_EVS_SetBinFltrMask command using a bit mask of '0x0000' which will result in getting all of the events for that Event Id or 2. You can remove the registration of that event with EVS (see \$sc_\$cpu_EVS_DelEvtFltr). Note that option (1) is the preferred method. 	
(Q) What is the purpose of DEBUG event messages?	
	Event messages 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.

Prev: [Effects of a Processor Reset on EVS](#)

Up To: [cFE Event Services Overview](#)

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

Global `CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE`

Default EVS Application Data Filename

Global `CFE_PLATFORM_EVS_DEFAULT_LOG_FILE`

Default Event Log Filename

Global `CFE_PLATFORM_EVS_DEFAULT_LOG_MODE`

Default EVS Local Event Log Mode

Global `CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE`

Default EVS Message Format Mode

Global `CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG`

Default EVS Event Type Filter Mask

Global `CFE_PLATFORM_EVS_LOG_MAX`

Maximum Number of Events in EVS Local Event Log

Global `CFE_PLATFORM_EVS_MAX_EVENT_FILTERS`

Define Maximum Number of Event Filters per Application

Global `CFE_PLATFORM_EVS_PORT_DEFAULT`

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](#)

Next: [Messages](#)

Up To: [cFE Software Bus Overview](#)

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.

Next: [Pipes](#)

Up To: [Software Bus Terminology](#)

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 Pipeld) 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](#).

Next: [Subscriptions](#)

Prev: [Messages](#)

Up To: [Software Bus Terminology](#)

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.

Next: [Memory](#)

Prev: [Pipes](#)

Up To: [Software Bus Terminology](#)

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.

Next: [Autonomous Actions](#)

Prev: [Subscriptions](#)

Up To: [Software Bus Terminology](#)

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.

Next: [Operation of the SB Software](#)

Prev: [Software Bus Terminology](#)

Up To: [cFE Software Bus Overview](#)

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](#)

Next: [Initialization](#)

Prev: [Autonomous Actions](#)

Up To: [cFE Software Bus Overview](#)

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.

Next: [All Resets](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Message Routing](#)

Prev: [Initialization](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Packet Sequence Values](#)

Prev: [All Resets](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Message Limit Error](#)

Prev: [Message Routing](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Pipe Overflow Error](#)

Prev: [Packet Sequence Values](#)

Up To: [Operation of the SB Software](#)

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](#).

Next: [SB Event Filtering](#)

Prev: [Message Limit Error](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Diagnostic Data](#)

Prev: [Pipe Overflow Error](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Control of Packet Routing](#)

Prev: [SB Event Filtering](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Quality of Service](#)

Prev: [Diagnostic Data](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Known Problem](#)

Prev: [Control of Packet Routing](#)

Up To: [Operation of the SB Software](#)

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.

Next: [Frequently Asked Questions about Software Bus](#)

Prev: [Quality of Service](#)

Up To: [Operation of the SB Software](#)

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?	
<p>The memory pool handle is used to analyze the SB memory pool statistics. The cFE ES command (CFE_ES_S_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</p>	
(Q) When sending a message, what message header fields are critical for routing the message?	
<p>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.</p>	
(Q) How many copies of the message are performed in a typical message delivery?	
<p>There is a single copy of the message performed when sending a message (from the callers memory space) using <code>CFE_SB_TransmitMsg</code>. 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 <code>CFE_SB_TransmitBuffer</code>. 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.</p>	
(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?	
<p>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.</p>	
(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.	
<p>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 call to CFE_SB_ReceiveBuffer, <code>Ptr</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.</p>	
(Q) Why am I not seeing expected Message Limit error events or Pipe Overflow events?	
<p>It is possible the events are being filtered by cFE Event Services. The filtering for this event may be specified in the platform configuration file or it may have been commanded after the system initializes. There is a corresponding counter for each of these conditions. First verify that the condition is happening by viewing the counter in SB HK telemetry. If the condition is happening, you can view the SB filter information through the EVS App Data Main page by clicking the 'go to' button for SB. The event Id for these events can be learned through a previous event or from the cfe_sb_events.h file.</p>	
(Q) Why does the SB provide event filtering through the platform configuration file?	
<p>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.</p>	
(Q) Why does SB have so many debug event messages?	
<p>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.</p>	
(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.

Prev: [Operation of the SB Software](#)

Up To: [cFE Software Bus Overview](#)

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

Global `CFE_MISSION_SB_MAX_SB_MSG_SIZE`

Maximum SB Message Size

Global `CFE_PLATFORM_ENDIAN`

Platform Endian Indicator

Global `CFE_PLATFORM_SB_BUF_MEMORY_BYTES`

Size of the SB buffer memory pool

Global `CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME`

Default Message Map Filename

Global `CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT`

Default Subscription Message Limit

Global `CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME`

Default Pipe Information Filename

Global CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME

Default Routing Information Filename

Global CFE_PLATFORM_SB_FILTERED_EVENT1

SB Event Filtering

Global CFE_PLATFORM_SB_HIGHEST_VALID_MSGID

Highest Valid Message Id

Global CFE_PLATFORM_SB_MAX_DEST_PER_PKT

Maximum Number of unique local destinations a single MsgId can have

Global CFE_PLATFORM_SB_MAX_MSG_IDS

Maximum Number of Unique Message IDs SB Routing Table can hold

Global CFE_PLATFORM_SB_MAX_PIPES

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.

Next: [cFE Table Types and Table Options](#)
Up To: [cFE Table Services Overview](#)

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

Next: [Single Buffered Tables](#)

Prev: [Managing Tables](#)

Up To: [cFE Table Services Overview](#)

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](#)).

Next: [Double Buffered Tables](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [Tables with Validation Functions](#)

Prev: [Single Buffered Tables](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [Critical Tables](#)

Prev: [Double Buffered Tables](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [User Defined Address Tables](#)

Prev: [Tables with Validation Functions](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [Dump Only Tables](#)

Prev: [Critical Tables](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [Table Registry](#)

Prev: [User Defined Address Tables](#)

Up To: [cFE Table Types and Table Options](#)

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.

Next: [Table Services Telemetry](#)
Prev: [cFE Table Types and Table Options](#)
Up To: [cFE Table Services Overview](#)

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](#).

Next: [Effects of Processor Reset on Tables](#)
Prev: [Table Registry](#)
Up To: [cFE Table Services Overview](#)

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.

Next: [Frequently Asked Questions about Table Services](#)
Prev: [Table Services Telemetry](#)
Up To: [cFE Table Services Overview](#)

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?	
	<p>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.</p> <p>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.</p>
(Q) I tried to validate a table and received the following event message that said the event failed:	
<pre>"MyApp validation failed for Inactive 'MyApp.MyTable', Status=0x####"</pre>	
What happened?	
	<p>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.</p>
(Q) What commands do I use to load a table with a new image?	
	<p>There are a number of steps required to load a table.</p> <ol style="list-style-type: none"> 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 <code>e1f2cfetbl</code> 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:	
<p><i>CFE_TBL:GetAddressInternal-App(%d) attempt to access unowned Tbl Handle=%d</i></p>	
	<p>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).</p>
(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.
3. A table image was loaded; the image was successfully validated; the command for activation was sent; but the application fails to perform the activation.

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

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

Prev: [Effects of Processor Reset on Tables](#)

Up To: [cFE Table Services Overview](#)

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

Global CFE_MISSION_TBL_MAX_NAME_LENGTH

Maximum Table Name Length

Global CFE_PLATFORM_TBL_BUF_MEMORY_BYTES

Size of Table Services Table Memory Pool

Global CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE

Default Filename for a Table Registry Dump

Global CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES

Maximum Number of Critical Tables that can be Registered

Global CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE

Maximum Size Allowed for a Double Buffered Table

Global CFE_PLATFORM_TBL_MAX_NUM_HANDLES

Maximum Number of Table Handles

Global CFE_PLATFORM_TBL_MAX_NUM_TABLES

Maximum Number of Tables Allowed to be Registered

Global CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS

Maximum Number of Simultaneous Table Validations

Global CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS

Maximum Number of Simultaneous Loads to Support

Global CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE

Maximum Size Allowed for a Single Buffered Table

Global CFE_PLATFORM_TBL_VALID_PRID_1

Processor ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_PRID_COUNT

Number of Processor ID's specified for validation

Global CFE_PLATFORM_TBL_VALID_SCID_1

Spacecraft ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_SCID_COUNT

Number of Spacecraft ID's specified for validation

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

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.

Next: [Time Structure](#)
Up To: [cFE Time Services Overview](#)

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;
```

Next: [Time Formats](#)

Prev: [Time Components](#)

Up To: [cFE Time Services Overview](#)

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:

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

Next: [Time Configuration](#)

Prev: [Time Components](#)

Up To: [cFE Time Services Overview](#)

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

Next: [Time Services Paradigm\(s\)](#)

Prev: [Time Formats](#)

Up To: [cFE Time Services Overview](#)

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](#)

Next: [Enabling Fake Tone Signal](#)

Up To: [Time Configuration](#)

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](#)

Next: [Selecting Tone and Data Ordering](#)

Prev: [Time Format Selection](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Tone and Data Window](#)

Prev: [Enabling Fake Tone Signal](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Server/Client](#)

Prev: [Selecting Tone and Data Ordering](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Tone Byte Order](#)

Prev: [Specifying Tone and Data Window](#)

Up To: [Time Configuration](#)

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
```

Next: [Virtual MET](#)

Prev: [Specifying Time Server/Client](#)

Up To: [Time Configuration](#)

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

Next: [Specifying Time Source](#)

Prev: [Specifying Time Tone Byte Order](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Signal](#)

Prev: [Virtual MET](#)

Up To: [Time Configuration](#)

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](#)

Next: [Time Services Paradigm\(s\)](#)

Prev: [Specifying Time Source](#)

Up To: [Time Configuration](#)

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](#)

Next: [Enabling Fake Tone Signal](#)

Up To: [Time Configuration](#)

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](#)

Next: [Selecting Tone and Data Ordering](#)

Prev: [Time Format Selection](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Tone and Data Window](#)

Prev: [Enabling Fake Tone Signal](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Server/Client](#)

Prev: [Selecting Tone and Data Ordering](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Tone Byte Order](#)

Prev: [Specifying Tone and Data Window](#)

Up To: [Time Configuration](#)

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
```

Next: [Virtual MET](#)

Prev: [Specifying Time Server/Client](#)

Up To: [Time Configuration](#)

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

Next: [Specifying Time Source](#)

Prev: [Specifying Time Tone Byte Order](#)

Up To: [Time Configuration](#)

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](#)

Next: [Specifying Time Signal](#)

Prev: [Virtual MET](#)

Up To: [Time Configuration](#)

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](#)

Next: [Time Services Paradigm\(s\)](#)

Prev: [Specifying Time Source](#)

Up To: [Time Configuration](#)

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.

Next: [Flywheeling](#)

Prev: [Time Components](#)

Up To: [cFE Time Services Overview](#)

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.

Next: [Time State](#)

Prev: [Time Services Paradigm\(s\)](#)

Up To: [cFE Time Services Overview](#)

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.

Next: [Initialization](#)

Prev: [Flywheeling](#)

Up To: [cFE Time Services Overview](#)

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](#)

Next: [Power-On Reset](#)

Prev: [Time State](#)

Up To: [cFE Time Services Overview](#)

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.

Next: [Processor Reset](#)

Up To: [Initialization](#)

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.

Next: [Normal Operation](#)

Prev: [Power-On Reset](#)

Up To: [Initialization](#)

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.

Next: [Processor Reset](#)

Up To: [Initialization](#)

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.

Next: [Normal Operation](#)

Prev: [Power-On Reset](#)

Up To: [Initialization](#)

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](#)

Next: [Power-On Reset](#)

Prev: [Time State](#)

Up To: [cFE Time Services Overview](#)

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.

Next: [Processor Reset](#)

Up To: [Initialization](#)

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.

Next: [Normal Operation](#)

Prev: [Power-On Reset](#)

Up To: [Initialization](#)

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.

Next: [Processor Reset](#)

Up To: [Initialization](#)

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.

Next: [Normal Operation](#)

Prev: [Power-On Reset](#)

Up To: [Initialization](#)

2.22.23 Normal Operation

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

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

Next: [Client](#)

Prev: [Initialization](#)

Up To: [cFE Time Services Overview](#)

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.

Next: [Server](#)

Up To: [Normal Operation](#)

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](#)

Next: [Setting Time](#)

Prev: [Client](#)

Up To: [Normal Operation](#)

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](#)

Next: [Adjusting Time](#)

Up To: [Server](#)

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_1HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_1HZ_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_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECOND_CC](#)

Next: [Setting MET](#)

Prev: [Setting Time](#)

Up To: [Server](#)

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](#)

Next: [Frequently Asked Questions](#)

Prev: [Adjusting Time](#)

Up To: [Server](#)

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.

Next: [Server](#)

Up To: [Normal Operation](#)

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](#)

Next: [Setting Time](#)

Prev: [Client](#)

Up To: [Normal Operation](#)

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](#)

Next: [Adjusting Time](#)

Up To: [Server](#)

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_1HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_1HZ_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_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECOND_CC](#)

Next: [Setting MET](#)

Prev: [Setting Time](#)

Up To: [Server](#)

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](#)

Next: [Frequently Asked Questions](#)

Prev: [Adjusting Time](#)

Up To: [Server](#)

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](#)

Next: [Adjusting Time](#)

Up To: [Server](#)

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_1HZ_ADJUSTMENT_CC](#) and [CFE_TIME_SUB_1HZ_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 TA-I-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

[CFE_TIME_ADD_ADJUST_CC](#), [CFE_TIME_SUB_ADJUST_CC](#), [CFE_TIME_SET_STCF_CC](#), [CFE_TIME_ADD_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SET_LEAP_SECOND_CC](#)

Next: [Setting MET](#)

Prev: [Setting Time](#)

Up To: [Server](#)

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](#)

Next: [Frequently Asked Questions](#)

Prev: [Adjusting Time](#)

Up To: [Server](#)

2.22.29 Frequently Asked Questions

(Q)

Prev: [Normal Operation](#)
Up To: [cFE Time Services Overview](#)

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_1HZ_ADJUSTMENT_CC`

Add Delta to Spacecraft Time Correlation Factor each 1Hz

Global `CFE_TIME_ADD_ADJUST_CC`

Add Delta to Spacecraft Time Correlation Factor

Global `CFE_TIME_ADD_DELAY_CC`

Add Time to Tone Time Delay

Global `CFE_TIME_NOOP_CC`

Time No-Op

Global `CFE_TIME_RESET_COUNTERS_CC`

Time Reset Counters

Global `CFE_TIME_SEND_DIAGNOSTIC_TLM_CC`

Request TIME Diagnostic Telemetry

Global `CFE_TIME_SET_LEAP_SECONDS_CC`

Set Leap Seconds

Global `CFE_TIME_SET_MET_CC`

Set Mission Elapsed Time

Global `CFE_TIME_SET_SIGNAL_CC`

Set Tone Signal Source

Global `CFE_TIME_SET_SOURCE_CC`

Set Time Source

Global `CFE_TIME_SET_STATE_CC`

Set Time State

Global `CFE_TIME_SET_STCF_CC`

Set Spacecraft Time Correlation Factor

Global CFE_TIME_SET_TIME_CC

Set Spacecraft Time

Global CFE_TIME_SUB_1HZ_ADJUSTMENT_CC

Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

Global CFE_TIME_SUB_ADJUST_CC

Subtract Delta from Spacecraft Time Correlation Factor

Global CFE_TIME_SUB_DELAY_CC

Subtract Time from Tone Time Delay

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

Global CFE_MISSION_TIME_CFG_DEFAULT_TAI

Default Time Format

Global CFE_MISSION_TIME_CFG_FAKE_TONE

Default Time Format

Global CFE_MISSION_TIME_DEF_MET_SECS

Default Time Values

Global CFE_MISSION_TIME_EPOCH_YEAR

Default EPOCH Values

Global CFE_MISSION_TIME_FS_FACTOR

Time File System Factor

Global CFE_MISSION_TIME_MIN_ELAPSED

Min and Max Time Elapsed

Global CFE_PLATFORM_TIME_CFG_LATCH_FLY

Define Periodic Time to Update Local Clock Tone Latch

Global CFE_PLATFORM_TIME_CFG_SERVER

Time Server or Time Client Selection

Global CFE_PLATFORM_TIME_CFG_SIGNAL

Include or Exclude the Primary/Redundant Tone Selection Cmd

Global CFE_PLATFORM_TIME_CFG_SOURCE

Include or Exclude the Internal/External Time Source Selection Cmd

Global CFE_PLATFORM_TIME_CFG_SRC_MET

Choose the External Time Source for Server only

Global CFE_PLATFORM_TIME_CFG_START_FLY

Define Time to Start Flywheel Since Last Tone

Global CFE_PLATFORM_TIME_CFG_TONE_LIMIT

Define Timing Limits From One Tone To The Next

Global CFE_PLATFORM_TIME_CFG_VIRTUAL

Time Tone In Big-Endian Order

Local MET or Virtual MET Selection for Time Servers

Global CFE_PLATFORM_TIME_MAX_DELTA_SECS

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

Global CFE_PLATFORM_TIME_MAX_LOCAL_SECS

Define the Local Clock Rollover Value in seconds and subseconds

Global CFE_PLATFORM_TIME_START_TASK_PRIORITY

Define TIME Task Priorities

Global CFE_PLATFORM_TIME_START_TASK_STACK_SIZE

Define TIME Task Stack Sizes

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_SYSLOG_CC

\$sc_\$cpu_ES_ClearSysLog

Global CFE_ES_DELETE_CDS_CC

\$sc_\$cpu_ES_DeleteCDS

Global CFE_ES_DUMP_CDS_REGISTRY_CC

\$sc_\$cpu_ES_WriteCDS2File

Global CFE_ES_NOOP_CC

\$sc_\$cpu_ES_NOOP

Global CFE_ES_OVER_WRITE_SYSLOG_CC

\$sc_\$cpu_ES_OverwriteSysLogMode

Global CFE_ES_QUERY_ALL_CC

\$sc_\$cpu_ES_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_SYSLOG_CC

\$sc_\$cpu_ES_WriteSysLog2File

Global CFE_EVS_ADD_EVENT_FILTER_CC

\$sc_\$cpu_EVS_AddEvtFltr

Global CFE_EVS_CLEAR_LOG_CC

\$sc_\$cpu_EVS_ClrLog

Global CFE_EVS_DELETE_EVENT_FILTER_CC

\$sc_\$cpu_EVS_DelEvtFltr

Global CFE_EVS_DISABLE_APP_EVENT_TYPE_CC

\$sc_\$cpu_EVS_DisAppEvtType, \$sc_\$cpu_EVS_DisAppEvtTypeMask

Global CFE_EVS_DISABLE_APP_EVENTS_CC

\$sc_\$cpu_EVS_DisAppEvGen

Global CFE_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_1HZ_ADJUSTMENT_CC
\$sc_\$cpu_TIME_Add1HzSTCF

Global CFE_TIME_ADD_ADJUST_CC
\$sc_\$cpu_TIME_AddSTCFAj

Global CFE_TIME_ADD_DELAY_CC
\$sc_\$cpu_TIME_AddClockLat

Global CFE_TIME_NOOP_CC
\$sc_\$cpu_TIME_NOOP

Global CFE_TIME_RESET_COUNTERS_CC
\$sc_\$cpu_TIME_ResetCtrs

Global CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
\$sc_\$cpu_TIME_RequestDiag

Global CFE_TIME_SET_LEAP_SECONDS_CC
\$sc_\$cpu_TIME_SetClockLeap

Global CFE_TIME_SET_MET_CC
\$sc_\$cpu_TIME_SetClockMET

Global CFE_TIME_SET_SIGNAL_CC
\$sc_\$cpu_TIME_SetSignal

Global CFE_TIME_SET_SOURCE_CC
\$sc_\$cpu_TIME_SetSource

Global CFE_TIME_SET_STATE_CC
\$sc_\$cpu_TIME_SetState

Global CFE_TIME_SET_STCF_CC
\$sc_\$cpu_TIME_SetClockSTCF

Global CFE_TIME_SET_TIME_CC
\$sc_\$cpu_TIME_SetClock

Global CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
\$sc_\$cpu_TIME_Sub1HzSTCF

Global CFE_TIME_SUB_ADJUST_CC
\$sc_\$cpu_TIME_SubSTCFAj

Global CFE_TIME_SUB_DELAY_CC
\$sc_\$cpu_TIME_SubClockLat

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::ResourceID

\$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_CFEMINORVER

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::PerfDataToWrite
\$sc_\$cpu_ES_PerfData2Write

Global CFE_ES_HousekeepingTlm_Payload::PerfFilterMask [CFE_MISSION_ES_PERF_MAX_IDS/32]
\$sc_\$cpu_ES_PerfFitrMask[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_PoolStatsTim_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::Padding
\$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].SPARE2ALIGN3

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_LOGOVERLOWC

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_EVS_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::GetPipeIDByNameErrorCounter
\$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::PipId
\$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_SMPPALW

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_SMPSBBIU

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_LastUpdTime, \$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::FileCreateTimeSecs
\$sc_\$cpu_TBL_FILECSECONDS

Global CFE_TBL_TblRegPacket_Payload::FileCreateTimeSubSecs
\$sc_\$cpu_TBL_FILECSUBSECONDS

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_DLapS

Global CFE_TIME_DiagnosticTlm_Payload::AtToneMET
\$sc_\$cpu_TIME_DTMETS, \$sc_\$cpu_TIME_DTMETSs

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_DAPISate

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_D1HzISRCNT

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_DAPISate

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.
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

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 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.

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.

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_BackgroundFileDumplIsPending](#) - Query if a background file write request is currently pending.

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.

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.

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](#).

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.

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_RegisterSynchCallback](#) - Registers a callback function that is called whenever time synchronization occurs.
 - [CFE_TIME_UnregisterSynchCallback](#) - Unregisters a callback function that is called whenever time synchronization occurs.
- [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_HK_TLM_MSG

cFE Portable Message Numbers for Telemetry

Global CFE_MISSION_EVS_CMD_MSG

cFE Portable Message Numbers for Commands

Global CFE_MISSION_MAX_API_LEN

cFE Maximum length for API names within data exchange structures

Global CFE_MISSION_MAX_FILE_LEN

cFE Maximum length for filenames within data exchange structures

Global CFE_MISSION_MAX_PATH_LEN

cFE Maximum length for pathnames within data exchange structures

Global CFE_MISSION_TIME_DATA_CMD_MSG

cFE Portable Message Numbers for Global Messages

7 Module Index

7.1 Modules

Here is a list of all modules:

CFS Health and Safety Mission Configuration	135
CFS Health and Safety Command Message IDs	137
CFS Health and Safety Telemetry Message IDs	138
CFS Health and Safety Platform Configuration	139
CFS Health and Safety Event IDs	159
CFS Health and Safety Command Structures	204
CFS Health and Safety Telemetry	205
CFS Health and Safety Command Codes	206
CFS Health and Safety Version	222
cFE Return Code Defines	223

cFE Resource ID APIs	260
cFE Entry/Exit APIs	264
cFE Application Control APIs	266
cFE Application Behavior APIs	269
cFE Information APIs	274
cFE Child Task APIs	284
cFE Miscellaneous APIs	289
cFE Critical Data Store APIs	293
cFE Memory Manager APIs	299
cFE Performance Monitor APIs	307
cFE Generic Counter APIs	310
cFE Registration APIs	317
cFE Send Event APIs	319
cFE Reset Event Filter APIs	324
cFE File Header Management APIs	326
cFE File Utility APIs	331
cFE Generic Message APIs	337
cFE Message Primary Header APIs	338
cFE Message Extended Header APIs	349
cFE Message Secondary Header APIs	356
cFE Message Id APIs	362
cFE Pipe Management APIs	365
cFE Message Subscription Control APIs	372
cFE Send/Receive Message APIs	378
cFE Zero Copy APIs	381
cFE Message Characteristics APIs	384
cFE Message ID APIs	389
cFE SB Pipe options	392
cFE Registration APIs	393
cFE Manage Table Content APIs	399

cFE Access Table Content APIs	406
cFE Get Table Information APIs	411
cFE Table Type Defines	415
cFE Get Current Time APIs	418
cFE Get Time Information APIs	422
cFE Time Arithmetic APIs	425
cFE Time Conversion APIs	428
cFE External Time Source APIs	431
cFE Miscellaneous Time APIs	436
cFE Resource ID base values	438
cFE Clock State Flag Defines	440
OSAL Semaphore State Defines	443
OSAL Binary Semaphore APIs	444
OSAL BSP low level access APIs	450
OSAL Real Time Clock APIs	451
OSAL Core Operation APIs	463
OSAL Counting Semaphore APIs	467
OSAL Directory APIs	473
OSAL Return Code Defines	478
OSAL Error Info APIs	489
OSAL File Access Option Defines	491
OSAL Reference Point For Seek Offset Defines	492
OSAL Standard File APIs	493
OSAL File System Level APIs	506
OSAL Heap APIs	515
OSAL Object Type Defines	516
OSAL Object ID Utility APIs	520
OSAL Dynamic Loader and Symbol APIs	527
OSAL Mutex APIs	532
OSAL Network ID APIs	537

OSAL Printf APIs	539
OSAL Message Queue APIs	541
OSAL Select APIs	546
OSAL Shell APIs	551
OSAL Socket Address APIs	552
OSAL Socket Management APIs	556
OSAL Task APIs	564
OSAL Time Base APIs	571
OSAL Timer APIs	577

8 Data Structure Index

8.1 Data Structures

Here are the data structures with brief descriptions:

CCSDS_ExtendedHeader CCSDS packet extended header	584
CCSDS_PrimaryHeader CCSDS packet primary header	585
CFE_ES_AppInfo Application Information	586
CFE_ES_AppNameCmd Generic application name command	592
CFE_ES_AppNameCmd_Payload Generic application name command payload	593
CFE_ES_AppReloadCmd_Payload Reload Application Command Payload	594
CFE_ES_BlockStats Block statistics	595
CFE_ES_CDSRegDumpRec CDS Register Dump Record	596
CFE_ES_DeleteCDSCmd Delete Critical Data Store Command	598
CFE_ES_DeleteCDSCmd_Payload Delete Critical Data Store Command Payload	599

CFE_ES_DumpCDSRegistryCmd	Dump CDS Registry Command	600
CFE_ES_DumpCDSRegistryCmd_Payload	Dump CDS Registry Command Payload	601
CFE_ES_FileNameCmd	Generic file name command	602
CFE_ES_FileNameCmd_Payload	Generic file name command payload	603
CFE_ES_HousekeepingTlm		604
CFE_ES_HousekeepingTlm_Payload		604
CFE_ES_MemPoolStats	Memory Pool Statistics	617
CFE_ES_MemStatsTlm		619
CFE_ES_NoArgsCmd	Generic "no arguments" command	620
CFE_ES_OneAppTlm		621
CFE_ES_OneAppTlm_Payload		622
CFE_ES_OverWriteSysLogCmd	Overwrite/Discard System Log Configuration Command Payload	623
CFE_ES_OverWriteSysLogCmd_Payload	Overwrite/Discard System Log Configuration Command Payload	624
CFE_ES_PoolAlign	Pool Alignment	625
CFE_ES_PoolStatsTlm_Payload		626
CFE_ES_ReloadAppCmd	Reload Application Command	627
CFE_ES_RestartCmd	Restart cFE Command	628
CFE_ES_RestartCmd_Payload	Restart cFE Command Payload	629
CFE_ES_SendMemPoolStatsCmd	Send Memory Pool Statistics Command	630
CFE_ES_SendMemPoolStatsCmd_Payload	Send Memory Pool Statistics Command Payload	631
CFE_ES_SetMaxPRCountCmd	Set Maximum Processor Reset Count Command	632

CFE_ES_SetMaxPRCountCmd_Payload	Set Maximum Processor Reset Count Command Payload	633
CFE_ES_SetPerfFilterMaskCmd	Set Performance Analyzer Filter Mask Command	633
CFE_ES_SetPerfFilterMaskCmd_Payload	Set Performance Analyzer Filter Mask Command Payload	634
CFE_ES_SetPerfTriggerMaskCmd	Set Performance Analyzer Trigger Mask Command	635
CFE_ES_SetPerfTrigMaskCmd_Payload	Set Performance Analyzer Trigger Mask Command Payload	636
CFE_ES_StartApp	Start Application Command	637
CFE_ES_StartAppCmd_Payload	Start Application Command Payload	638
CFE_ES_StartPerfCmd_Payload	Start Performance Analyzer Command Payload	640
CFE_ES_StartPerfDataCmd	Start Performance Analyzer Command	641
CFE_ES_StopPerfCmd_Payload	Stop Performance Analyzer Command Payload	642
CFE_ES_StopPerfDataCmd	Stop Performance Analyzer Command	643
CFE_ES_TaskInfo	Task Information	644
CFE_EVS_AppDataCmd_Payload	Write Event Services Application Information to File Command Payload	647
CFE_EVS_AppNameBitMaskCmd	Generic App Name and Bitmask Command	647
CFE_EVS_AppNameBitMaskCmd_Payload	Generic App Name and Bitmask Command Payload	648
CFE_EVS_AppNameCmd	Generic App Name Command	650
CFE_EVS_AppNameCmd_Payload	Generic App Name Command Payload	651
CFE_EVS_AppNameEventIDCmd	Generic App Name and Event ID Command	652
CFE_EVS_AppNameEventIDCmd_Payload	Generic App Name and Event ID Command Payload	653

CFE_EVS_AppNameEventIDMaskCmd Generic App Name, Event ID, Mask Command	654
CFE_EVS_AppNameEventIDMaskCmd_Payload Generic App Name, Event ID, Mask Command Payload	655
CFE_EVS_AppTlmData	656
CFE_EVS_BinFilter Event message filter definition structure	657
CFE_EVS_BitMaskCmd Generic Bitmask Command	658
CFE_EVS_BitMaskCmd_Payload Generic Bitmask Command Payload	659
CFE_EVS_HousekeepingTlm	660
CFE_EVS_HousekeepingTlm_Payload	661
CFE_EVS_LogFileCmd_Payload Write Event Log to File Command Payload	666
CFE_EVS_LongEventTlm	667
CFE_EVS_LongEventTlm_Payload	668
CFE_EVS_NoArgsCmd Command with no additional arguments	670
CFE_EVS_PacketID	670
CFE_EVS_SetEventFormatCode_Payload Set Event Format Mode Command Payload	672
CFE_EVS_SetEventFormatModeCmd Set Event Format Mode Command	673
CFE_EVS_SetLogMode_Payload Set Log Mode Command Payload	674
CFE_EVS_SetLogModeCmd Set Log Mode Command	675
CFE_EVS_ShortEventTlm	676
CFE_EVS_ShortEventTlm_Payload	677
CFE_EVS_WriteAppDataFileCmd Write Event Services Application Information to File Command	678
CFE_EVS_WriteLogFileCmd Write Event Log to File Command	679
CFE_FS_FileWriteMetaData External Metadata/State object associated with background file writes	680

CFE_FS_Header	Standard cFE File header structure definition	682
CFE_SB_AllSubscriptionsTlm		684
CFE_SB_AllSubscriptionsTlm_Payload		685
CFE_SB_HousekeepingTlm		687
CFE_SB_HousekeepingTlm_Payload		688
CFE_SB_Msg	Software Bus generic message	693
CFE_SB_MsgId_t	CFE_SB_MsgId_t type definition	694
CFE_SB_MsgMapFileEntry	SB Map File Entry	695
CFE_SB_PipeDepthStats	SB Pipe Depth Statistics	696
CFE_SB_PipeInfoEntry	SB Pipe Information File Entry	698
CFE_SB_Qos_t	Quality Of Service Type Definition	701
CFE_SB_RouteCmd	Enable/Disable Route Command	702
CFE_SB_RouteCmd_Payload	Enable/Disable Route Command Payload	703
CFE_SB_RoutingFileEntry	SB Routing File Entry	705
CFE_SB_SingleSubscriptionTlm		707
CFE_SB_SingleSubscriptionTlm_Payload		707
CFE_SB_StatsTlm		709
CFE_SB_StatsTlm_Payload		710
CFE_SB_SubEntries	SB Previous Subscriptions Entry	715
CFE_SB_WriteFileInfoCmd	Write File Info Command	717
CFE_SB_WriteFileInfoCmd_Payload	Write File Info Command Payload	718
CFE_TBL_AbortLoadCmd	Abort Load Command	718

CFE_TBL_AbortLoadCmd_Payload		
Abort Load Command Payload		719
CFE_TBL_ActivateCmd		
Activate Table Command		720
CFE_TBL_ActivateCmd_Payload		
Activate Table Command Payload		721
CFE_TBL_DeleteCDSCmd_Payload		
Delete Critical Table CDS Command Payload		722
CFE_TBL_DeleteCDSCmd		
Delete Critical Table CDS Command		723
CFE_TBL_DumpCmd		
CFE_TBL_DumpCmd_Payload		
Dump Table Command Payload		725
CFE_TBL_DumpRegistryCmd		
Dump Registry Command		726
CFE_TBL_DumpRegistryCmd_Payload		
Dump Registry Command Payload		727
CFE_TBL_File_Hdr		
The definition of the header fields that are included in CFE Table Data files		728
CFE_TBL_FileDef		
CFE_TBL_HousekeepingTIm		
CFE_TBL_HousekeepingTIm_Payload		
CFE_TBL_Info		
Table Info		738
CFE_TBL_LoadCmd		
Load Table Command		742
CFE_TBL_LoadCmd_Payload		
Load Table Command Payload		743
CFE_TBL_NoArgsCmd		
Generic "no arguments" command		744
CFE_TBL_NotifyCmd		
CFE_TBL_NotifyCmd_Payload		
Table Management Notification Command Payload		745
CFE_TBL_SendRegistryCmd		
Send Table Registry Command		746
CFE_TBL_SendRegistryCmd_Payload		
Send Table Registry Command Payload		747

CFE_TBL_TableRegistryTlm	748
CFE_TBL_TblRegPacket_Payload	749
CFE_TBL_ValidateCmd Validate Table Command	755
CFE_TBL_ValidateCmd_Payload Validate Table Command Payload	756
CFE_TIME_DiagnosticTlm	757
CFE_TIME_DiagnosticTlm_Payload	757
CFE_TIME_HousekeepingTlm	770
CFE_TIME_HousekeepingTlm_Payload	771
CFE_TIME_LeapsCmd_Payload Set leap seconds command payload	775
CFE_TIME_NoArgsCmd Generic no argument command	776
CFE_TIME_OneHzAdjustmentCmd Generic seconds, subseconds adjustment command	777
CFE_TIME_OneHzAdjustmentCmd_Payload Generic seconds, subseconds command payload	778
CFE_TIME_SetLeapSecondsCmd Set leap seconds command	779
CFE_TIME_SetSignalCmd Set tone signal source command	780
CFE_TIME_SetSourceCmd Set time data source command	781
CFE_TIME_SetStateCmd Set clock state command	782
CFE_TIME_SignalCmd_Payload Set tone signal source command payload	783
CFE_TIME_SourceCmd_Payload Set time data source command payload	784
CFE_TIME_StateCmd_Payload Set clock state command payload	784
CFE_TIME_SysTime Data structure used to hold system time values	785
CFE_TIME_TimeCmd Generic seconds, microseconds argument command	786

CFE_TIME_TimeCmd_Payload	Generic seconds, microseconds command payload	787
CFE_TIME_ToneDataCmd	Time at tone data command	788
CFE_TIME_ToneDataCmd_Payload	Time at tone data command payload	789
HS_AMTEntry_t	Application Monitor Table (AMT) Entry	791
HS_AppData_t	HS Global Data Structure	792
HS_CDSData_t	HS CDS Data Structure	804
HS_CustomData_t	HS custom global structure	806
HS_EMTEEntry_t	Event Monitor Table (EMT) Entry	810
HS_HkPacket_t	Housekeeping Packet Structure	812
HS_MATEEntry_t	Message Actions Table (MAT) Entry	817
HS_MATMsgBuf_t	Message Action Table buffer	818
HS_NoArgsCmd_t	No Arguments Command	819
HS_SetMaxResetsCmd_t	Set Max Resets Command	820
HS_SetUtilDiagCmd_t	Set Utilization Diagnostics Command	821
HS_SetUtilParamsCmd_t	Set Utililizition Parameters Command	822
HS_XCTEntry_t	Execution Counters Table (XCT) Entry	824
OS_bin_sem_prop_t	OSAL binary semaphore properties	825
OS_count_sem_prop_t	OSAL counting semaphore properties	826
os_dirent_t	Directory entry	827

OS_FdSet	An abstract structure capable of holding several OSAL IDs	828
OS_file_prop_t	OSAL file properties	829
os_fsinfo_t	OSAL file system info	830
os_fstat_t	File system status	831
OS_heap_prop_t	OSAL heap properties	833
OS_module_address_t	OSAL module address properties	834
OS_module_prop_t	OSAL module properties	836
OS_mut_sem_prop_t	OSAL mutex properties	837
OS_queue_prop_t	OSAL queue properties	838
OS_SockAddr_t	Encapsulates a generic network address	839
OS_SockAddrData_t	Storage buffer for generic network address	840
OS_socket_prop_t	Encapsulates socket properties	841
OS_static_symbol_record_t	Associates a single symbol name with a memory address	842
OS_statvfs_t		843
OS_task_prop_t	OSAL task properties	845
OS_time_t	OSAL time interval structure	846
OS_timebase_prop_t	Time base properties	847
OS_timer_prop_t	Timer properties	848

9.1 File List

Here is a list of all files with brief descriptions:

apps/hs/fsw/mission_inc/hs_perfids.h	850
apps/hs/fsw/platform_inc/hs_msgids.h	850
apps/hs/fsw/platform_inc/hs_platform_cfg.h	851
apps/hs/fsw/src/hs_app.c	853
apps/hs/fsw/src/hs_app.h	865
apps/hs/fsw/src/hs_cmds.c	878
apps/hs/fsw/src/hs_cmds.h	898
apps/hs/fsw/src/hs_custom.c	918
apps/hs/fsw/src/hs_custom.h	929
apps/hs/fsw/src/hs_events.h	942
apps/hs/fsw/src/hs_monitors.c	945
apps/hs/fsw/src/hs_monitors.h	954
apps/hs/fsw/src/hs_msg.h	964
apps/hs/fsw/src/hs_msgdefs.h	965
apps/hs/fsw/src/hs_tbl.h	968
apps/hs/fsw/src/hs_tbldefs.h	969
apps/hs/fsw/src/hs_utils.c	980
apps/hs/fsw/src/hs_utils.h	984
apps/hs/fsw/src/hs_verify.h	987
apps/hs/fsw/src/hs_version.h	987
apps/hs/fsw/tables/hs_amt.c	988
apps/hs/fsw/tables/hs_emt.c	989
apps/hs/fsw/tables/hs_mat.c	990
apps/hs/fsw/tables/hs_xct.c	991
build/docs/osconfig-example.h	992
cfe/cmake/sample_defs/cpu1_msgids.h	1000
cfe/cmake/sample_defs/cpu1_platform_cfg.h	1009

cfe/cmake/sample_defs/ sample_mission_cfg.h	1066
cfe/cmake/sample_defs/ sample_perfids.h	1087
cfe/modules/core_api/fsw/inc/cfe.h	1091
cfe/modules/core_api/fsw/inc/cfe_config.h	1092
cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h	1096
cfe/modules/core_api/fsw/inc/cfe_endian.h	1098
cfe/modules/core_api/fsw/inc/cfe_error.h	1098
cfe/modules/core_api/fsw/inc/cfe_es.h	1107
cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h	1111
cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h	1118
cfe/modules/core_api/fsw/inc/cfe_evs.h	1130
cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h	1132
cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h	1135
cfe/modules/core_api/fsw/inc/cfe_fs.h	1140
cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h	1141
cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h	1144
cfe/modules/core_api/fsw/inc/cfe_msg.h	1147
cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h	1149
cfe/modules/core_api/fsw/inc/cfe_resourceid.h	1157
cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h	1164
cfe/modules/core_api/fsw/inc/cfe_sb.h	1166
cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h	1169
cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h	1173
cfe/modules/core_api/fsw/inc/cfe_tbl.h	1177
cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h	1178
cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h	1181
cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h	1183
cfe/modules/core_api/fsw/inc/cfe_time.h	1185
cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h	1187
cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h	1189

cfe/modules/core_api/fsw/inc/cfe_version.h	1195
cfe/modules/es/fsw/inc/cfe_es_events.h	1198
cfe/modules/es/fsw/inc/cfe_es_msg.h	1231
cfe/modules/evs/fsw/inc/cfe_evs_events.h	1267
cfe/modules/evs/fsw/inc/cfe_evs_msg.h	1281
cfe/modules/msg/fsw/inc/ccsds_hdr.h	1313
cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h	1314
cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h	1315
cfe/modules/sb/fsw/inc/cfe_sb_events.h	1317
cfe/modules/sb/fsw/inc/cfe_sb_msg.h	1341
cfe/modules/tbl/fsw/inc/cfe_tbl_events.h	1359
cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h	1385
cfe/modules/time/fsw/inc/cfe_time_events.h	1402
cfe/modules/time/fsw/inc/cfe_time_msg.h	1416
osal/src/os/inc/common_types.h	1442
osal/src/os/inc/osapi-binsem.h	1450
osal/src/os/inc/osapi-bsp.h	1451
osal/src/os/inc/osapi-clock.h	1452
osal/src/os/inc/osapi-common.h	1454
osal/src/os/inc/osapi-constants.h	1457
osal/src/os/inc/osapi-countsem.h	1458
osal/src/os/inc/osapi-dir.h	1459
osal/src/os/inc/osapi-error.h	1460
osal/src/os/inc/osapi-file.h	1463
osal/src/os/inc/osapi-filesystem.h	1467
osal/src/os/inc/osapi-heap.h	1469
osal/src/os/inc/osapi-idmap.h	1469
osal/src/os/inc/osapi-macros.h	1471
osal/src/os/inc/osapi-module.h	1473
osal/src/os/inc/osapi-mutex.h	1475

osal/src/os/inc/osapi-network.h	1476
osal/src/os/inc/osapi-printf.h	1476
osal/src/os/inc/osapi-queue.h	1477
osal/src/os/inc/osapi-select.h	1477
osal/src/os/inc/osapi-shell.h	1479
osal/src/os/inc/osapi-sockets.h	1479
osal/src/os/inc/osapi-task.h	1482
osal/src/os/inc/osapi-timebase.h	1485
osal/src/os/inc/osapi-timer.h	1486
osal/src/os/inc/osapi-version.h	1487
osal/src/os/inc/osapi.h	1493
psp/fsw/inc/cfe_psp.h	1493

10 Module Documentation

10.1 CFS Health and Safety Mission Configuration

Macros

- `#define HS_IDLETASK_PERF_ID 37`
Idle task performance ID.
- `#define HS_APPMAIN_PERF_ID 40`
Main task performance ID.

10.1.1 Detailed Description

10.1.2 Macro Definition Documentation

10.1.2.1 HS_APPMAIN_PERF_ID

```
#define HS_APPMAIN_PERF_ID 40
```

Main task performance ID.

Definition at line 33 of file hs_perfids.h.

Referenced by HS_AppMain().

10.1.2.2 HS_IDLETASK_PERF_ID

```
#define HS_IDLETASK_PERF_ID 37
```

Idle task performance ID.

Definition at line 32 of file hs_perfids.h.

Referenced by HS_IdleTask().

10.2 CFS Health and Safety Command Message IDs

Macros

- `#define HS_CMD_MID 0x18AE`
Msg ID for cmds to HS.
- `#define HS_SEND_HK_MID 0x18AF`
Msg ID to request HS housekeeping.
- `#define HS_WAKEUP_MID 0x18B0`
Msg ID to wake up HS.

10.2.1 Detailed Description

10.2.2 Macro Definition Documentation

10.2.2.1 HS_CMD_MID

```
#define HS_CMD_MID 0x18AE
```

Msg ID for cmds to HS.

Definition at line 32 of file hs_msgids.h.

Referenced by HS_AppPipe(), and HS_SbInit().

10.2.2.2 HS_SEND_HK_MID

```
#define HS_SEND_HK_MID 0x18AF
```

Msg ID to request HS housekeeping.

Definition at line 33 of file hs_msgids.h.

Referenced by HS_AppPipe(), HS_SbInit(), and HS_VerifyMsgLength().

10.2.2.3 HS_WAKEUP_MID

```
#define HS_WAKEUP_MID 0x18B0
```

Msg ID to wake up HS.

Definition at line 34 of file hs_msgids.h.

Referenced by HS_SbInit().

10.3 CFS Health and Safety Telemetry Message IDs

Macros

- #define HS_HK_TLM_MID 0x08AD

HS Housekeeping Telemetry.

10.3.1 Detailed Description

10.3.2 Macro Definition Documentation

10.3.2.1 HS_HK_TLM_MID

```
#define HS_HK_TLM_MID 0x08AD
```

HS Housekeeping Telemetry.

Definition at line 43 of file hs_msgids.h.

Referenced by HS_SblInit().

10.4 CFS Health and Safety Platform Configuration

Macros

- `#define HS_APP_NAME "HS"`
Application Name.
- `#define HS_IDLE_TASK_NAME "HS_IDLE_TASK"`
Idle Task Configuration Parameters (custom)
- `#define HS_IDLE_TASK_STACK_PTR 0`
- `#define HS_IDLE_TASK_STACK_SIZE 4096`
- `#define HS_IDLE_TASK_FLAGS 0`
- `#define HS_IDLE_TASK_PRIORITY 252`
Idle Task Priority (custom)
- `#define HS_MAX_EXEC_CNT_SLOTS 32`
Maximum reported execution counters.
- `#define HS_MAX_MSG_ACT_TYPES 8`
Maximum message action types.
- `#define HS_MAX_MSG_ACT_SIZE 16`
Maximum message action size (in bytes)
- `#define HS_MAX_MONITORED_APPS 32`
Maximum number of monitored applications.
- `#define HS_MAX_MONITORED_EVENTS 16`
Maximum number of monitored events.
- `#define HS_WATCHDOG_TIMEOUT_VALUE 10000`
Watchdog Timeout Value.
- `#define HS_POST_PROCESSING_DELAY 0`
Time to wait after performing processing (in milliseconds)
- `#define HS_WAKEUP_TIMEOUT 1200`
Wakeup Message Software Bus Timeout.
- `#define HS_CPU_ALIVE_STRING "."`
CPU aliveness output string.
- `#define HS_CPU_ALIVE_PERIOD 5`
CPU aliveness output period.
- `#define HS_MAX_RESTART_ACTIONS 3`
Max Number of Processor Resets that may be performed by HS.
- `#define HS_CMD_PIPE_DEPTH 12`
Software bus command pipe depth.
- `#define HS_EVENT_PIPE_DEPTH 32`
Software bus event pipe depth.
- `#define HS_WAKEUP_PIPE_DEPTH 1`
Software bus wakeup pipe depth.
- `#define HS_RESET_TASK_DELAY 50`
Time to wait before a processor reset (in milliseconds)
- `#define HS_STARTUP_SYNC_TIMEOUT 65000`
Time to wait for all apps to be started (in milliseconds)
- `#define HS_APPMON_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Application Monitor.

- `#define HS_EVENTMON_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Event Monitor.
- `#define HS_ALIVENESS_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Aliveness Indicator.
- `#define HS_CPUHOG_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the CPU Hogging Indicator.
- `#define HS_AMT_FILENAME "/cf/hs_amt.tbl"`
Application Monitor Table (AMT) filename.
- `#define HS_EMT_FILENAME "/cf/hs_emt.tbl"`
Event Monitor Table (EMT) filename.
- `#define HS_XCT_FILENAME "/cf/hs_xct.tbl"`
Execution Counter Table (XCT) filename.
- `#define HS_MAT_FILENAME "/cf/hs_mat.tbl"`
Message Actions Table (MAT) filename.
- `#define HS_UTIL_CALLS_PER_MARK 1`
CPU Utilization Calls per Mark (custom)
- `#define HS_UTIL_CYCLES_PER_INTERVAL 1`
CPU Utilization Cycles per Interval (custom)
- `#define HS_UTIL_PER_INTERVAL_TOTAL 10000`
CPU Utilization Total Utils Per Interval.
- `#define HS_UTIL_PER_INTERVAL_HOGGING 9900`
CPU Utilization Hogging Utils Per Interval.
- `#define HS_UTIL_CONV_MULT1 1`
CPU Utilization Conversion Factor Multiplication 1 (custom)
- `#define HS_UTIL_CONV_DIV 10000`
CPU Utilization Conversion Factor Division (custom)
- `#define HS_UTIL_CONV_MULT2 1`
CPU Utilization Conversion Factor Multiplication 2 (custom)
- `#define HS_UTIL_HOGGING_TIMEOUT 5`
CPU Utilization Hogging Timeout.
- `#define HS_UTIL_PEAK_NUM_INTERVAL 64`
CPU Peak Utilization Number of Intervals.
- `#define HS_UTIL_AVERAGE_NUM_INTERVAL 4`
CPU Average Utilization Number of Intervals.
- `#define HS_UTIL_DIAG_MASK 0xFFFFFFFF`
CPU Utilization Diagnostics Mask (custom)
- `#define HS_UTIL_TIME_DIAG_ARRAY_POWER 4`
CPU Utilization Diagnostics Array Configuration (custom)
- `#define HS_UTIL_TIME_DIAG_ARRAY_LENGTH (1 << (HS_UTIL_TIME_DIAG_ARRAY_POWER))`
- `#define HS_UTIL_TIME_DIAG_ARRAY_MASK (HS_UTIL_TIME_DIAG_ARRAY_LENGTH - 1)`
- `#define HS_MISSION_REV 0`
Mission specific version number for HS application.

10.4.1 Detailed Description

10.4.2 Macro Definition Documentation

10.4.2.1 HS_ALIVENESS_DEFAULT_STATE

```
#define HS_ALIVENESS_DEFAULT_STATE HS_STATE_ENABLED
```

Default State of the Aliveness Indicator.

Description:

State the Aliveness Indicator is set to when the HS application starts.

Limits:

Must be HS_STATE_ENABLED or HS_STATE_DISABLED

Definition at line 394 of file hs_platform_cfg.h.

Referenced by HS_ApplInit().

10.4.2.2 HS_AMT_FILENAME

```
#define HS_AMT_FILENAME "/cf/hs_amt.tbl"
```

Application Monitor Table (AMT) filename.

Description:

Default file to load the Applications Monitor Table from during a power-on reset sequence

Limits:

This string shouldn't be longer than [OS_MAX_PATH_LEN](#) for the target platform in question

Definition at line 419 of file hs_platform_cfg.h.

Referenced by HS_TblInit().

10.4.2.3 HS_APP_NAME

```
#define HS_APP_NAME "HS"
```

Application Name.

Description:

This definition must match the name used at startup by the cFE Executive Services when creating the HS application. Note that application names are also an argument to certain cFE commands. For example, the application name is needed to access tables via cFE Table Services commands.

Limits:

HS requires that this name be defined, but otherwise places no limits on the definition. Refer to CFE Executive Services for specific information on limits related to application names.

Definition at line 47 of file hs_platform_cfg.h.

10.4.2.4 HS_APPMON_DEFAULT_STATE

```
#define HS_APPMON_DEFAULT_STATE HS_STATE_ENABLED
```

Default State of the Application Monitor.

Description:

State the Application Monitor is set to when the HS application starts.

Limits:

Must be HS_STATE_ENABLED or HS_STATE_DISABLED

Definition at line 370 of file hs_platform_cfg.h.

Referenced by HS_ApplInit().

10.4.2.5 HS_CMD_PIPE_DEPTH

```
#define HS_CMD_PIPE_DEPTH 12
```

Software bus command pipe depth.

Description:

Depth of the software bus pipe HS uses for commands and HK requests. Used during initialization in the call to [CFE_SB_CreatePipe](#)

Limits:

This parameter must be greater than 0.

Definition at line 300 of file hs_platform_cfg.h.

Referenced by HS_SblInit().

10.4.2.6 HS_CPU_ALIVE_PERIOD

```
#define HS_CPU_ALIVE_PERIOD 5
```

CPU aliveness output period.

Description:

Rate in number of HS cycles at which the HS_CPU_ALIVE_STRING is output via the UART.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 268 of file hs_platform_cfg.h.

Referenced by HS_ProcessMain().

10.4.2.7 HS_CPU_ALIVE_STRING

```
#define HS_CPU_ALIVE_STRING ". "
```

CPU aliveness output string.

Description:

String that is output to via [OS_printf](#) periodically if aliveness is enabled.

Limits:

None.

Definition at line 254 of file hs_platform_cfg.h.

Referenced by HS_ProcessMain().

10.4.2.8 HS_CPUHOG_DEFAULT_STATE

```
#define HS_CPUHOG_DEFAULT_STATE HS_STATE_ENABLED
```

Default State of the CPU Hogging Indicator.

Description:

State the CPU Hogging Event Message is set to when the HS application starts.

Limits:

Must be HS_STATE_ENABLED or HS_STATE_DISABLED

Definition at line 406 of file hs_platform_cfg.h.

Referenced by HS_AppInit().

10.4.2.9 HS_EMT_FILENAME

```
#define HS_EMT_FILENAME "/cf/hs_emt.tbl"
```

Event Monitor Table (EMT) filename.

Description:

Default file to load the Event Monitor Table from during a power-on reset sequence

Limits:

This string shouldn't be longer than [OS_MAX_PATH_LEN](#) for the target platform in question

Definition at line 432 of file hs_platform_cfg.h.

Referenced by HS_TblInit().

10.4.2.10 HS_EVENT_PIPE_DEPTH

```
#define HS_EVENT_PIPE_DEPTH 32
```

Software bus event pipe depth.

Description:

Depth of the software bus pipe HS uses for event monitoring. This should be set to supply sufficient room for the expected event message load per second. Used during initialization in the call to [CFE_SB_CreatePipe](#)

Limits:

This parameter must be greater than 0.

Definition at line 314 of file hs_platform_cfg.h.

Referenced by HS_AppMain(), HS_EnableEventMonCmd(), and HS_SbInit().

10.4.2.11 HS_EVENTMON_DEFAULT_STATE

```
#define HS_EVENTMON_DEFAULT_STATE HS_STATE_ENABLED
```

Default State of the Event Monitor.

Description:

State the Event Monitor is set to when the HS application starts.

Limits:

Must be HS_STATE_ENABLED or HS_STATE_DISABLED

Definition at line 382 of file hs_platform_cfg.h.

Referenced by HS_AppInit().

10.4.2.12 HS_IDLE_TASK_FLAGS

```
#define HS_IDLE_TASK_FLAGS 0
```

Definition at line 63 of file hs_platform_cfg.h.

Referenced by HS_CustomInit().

10.4.2.13 HS_IDLE_TASK_NAME

```
#define HS_IDLE_TASK_NAME "HS_IDLE_TASK"
```

Idle Task Configuration Parameters (custom)

Description:

These parameters are used by [CFE_ES_CreateChildTask](#). Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

These limits will vary by platform and available resources.

Definition at line 60 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.14 HS_IDLE_TASK_PRIORITY

```
#define HS_IDLE_TASK_PRIORITY 252
```

Idle Task Priority (custom)

Description:

This parameter is used to set the priority of the Idle Task. It should be higher than all other user created tasks, but may need to be set lower than the maximum value if an OS uses its own minimum priority task. Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

This parameter can't be larger than 255.

Definition at line 78 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.15 HS_IDLE_TASK_STACK_PTR

```
#define HS_IDLE_TASK_STACK_PTR 0
```

Definition at line 61 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.16 HS_IDLE_TASK_STACK_SIZE

```
#define HS_IDLE_TASK_STACK_SIZE 4096
```

Definition at line 62 of file hs_platform_cfg.h.

Referenced by HS_CustomInit().

10.4.2.17 HS_MAT_FILENAME

```
#define HS_MAT_FILENAME "/cf/hs_mat.tbl"
```

Message Actions Table (MAT) filename.

Description:

Default file to load the Message Actions Table from during a power-on reset sequence

Limits:

This string shouldn't be longer than [OS_MAX_PATH_LEN](#) for the target platform in question

Definition at line 458 of file hs_platform_cfg.h.

Referenced by HS_TblInit().

10.4.2.18 HS_MAX_EXEC_CNT_SLOTS

```
#define HS_MAX_EXEC_CNT_SLOTS 32
```

Maximum reported execution counters.

Description:

Maximum number of execution counters that can be specified to be reported in telemetry.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This parameter will dictate the size of the Execution Counter Table (XCT):

XCT Size = HS_MAX_EXEC_CNT_SLOTS * sizeof([HS_XCTEntry_t](#))

The total size of this table should not exceed the cFE size limit for a single buffered table set by the [CFE_PLATFOR](#)
[M_TBL_MAX_SNGL_TABLE_SIZE](#) parameter

Definition at line 100 of file hs_platform_cfg.h.

Referenced by HS_HousekeepingReq(), HS_TblInit(), and HS_ValidateEMTable().

10.4.2.19 HS_MAX_MONITORED_APPS

```
#define HS_MAX_MONITORED_APPS 32
```

Maximum number of monitored applications.

Description:

Maximum number of applications that can be monitored to assure check-ins

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This parameter must be greater than 0.

This parameter will dictate the size of the Application Monitor Table (AMT):

AMT Size = HS_MAX_MONITORED_APPS * sizeof([HS_AMTEntry_t](#))

The total size of this table should not exceed the cFE size limit for a single buffered table set by the [CFE_PLATFORM_MAX_TBL_MAX_SNGL_TABLE_SIZE](#) parameter

Definition at line 170 of file hs_platform_cfg.h.

Referenced by HS_AppMonStatusRefresh(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_TblInit(), and HS_ValidateAMTable().

10.4.2.20 HS_MAX_MONITORED_EVENTS

```
#define HS_MAX_MONITORED_EVENTS 16
```

Maximum number of monitored events.

Description:

Maximum number of events that can be monitored

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This parameter must be greater than 0.

This parameter will dictate the size of the Event Monitor Table (EMT):

EMT Size = HS_MAX_MONITORED_EVENTS * sizeof([HS_EMTEEntry_t](#))

The total size of this table should not exceed the cFE size limit for a single buffered table set by the [CFE_PLATFORM_MAX_TBL_MAX_SNGL_TABLE_SIZE](#) parameter

Definition at line 194 of file hs_platform_cfg.h.

Referenced by HS_HousekeepingReq(), HS_MonitorEvent(), HS_TblInit(), and HS_ValidateEMTable().

10.4.2.21 HS_MAX_MSG_ACT_SIZE

```
#define HS_MAX_MSG_ACT_SIZE 16
```

Maximum message action size (in bytes)

Description:

Size in bytes of maximum length of software bus message that can be sent using a Message Action action type.

Limits:

This parameter can't be larger than [CFE_MISSION_SB_MAX_SB_MSG_SIZE](#)

This parameter can't be smaller than a packet header

This parameter will influence the size of the Message Action Table (MAT):

MAT Size = HS_MAX_MSG_ACT_TYPES * (HS_MAX_MSG_ACT_SIZE + 4)

The total size of this table should not exceed the cFE size limit for a single buffered table set by the [CFE_PLATFOR←M_TBL_MAX_SNGL_TABLE_SIZE](#) parameter

Definition at line 146 of file hs_platform_cfg.h.

10.4.2.22 HS_MAX_MSG_ACT_TYPES

```
#define HS_MAX_MSG_ACT_TYPES 8
```

Maximum message action types.

Description:

Maximum number of Message Action action types.

Limits:

This parameter can't be larger than 4 less than an unsigned 16 bit integer (65531).

This parameter must be greater than 0.

This parameter will influence the size of the Message Action Table (MAT):

MAT Size = HS_MAX_MSG_ACT_TYPES * (HS_MAX_MSG_ACT_SIZE + 4)

The total size of this table should not exceed the cFE size limit for a single buffered table set by the [CFE_PLATFOR←M_TBL_MAX_SNGL_TABLE_SIZE](#) parameter

Definition at line 123 of file hs_platform_cfg.h.

Referenced by HS_AMTActionIsValid(), HS_EMTActionIsValid(), HS_MonitorApplications(), HS_MonitorEvent(), HS_←MsgActsStatusRefresh(), HS_ProcessMain(), HS_TblInit(), and HS_ValidateMATable().

10.4.2.23 HS_MAX_RESTART_ACTIONS

```
#define HS_MAX_RESTART_ACTIONS 3
```

Max Number of Processor Resets that may be performed by HS.

Description:

Maximum number of times that the HS App will attempt a processor reset as the result of either an Application Monitor or Event Monitor Failure

Limits:

This parameter can't be larger than an unsigned 16 bit integer (65535).

Although not enforced, if this parameter is greater than or equal to [CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS](#) then a POWER-ON reset will occur before the max count is reached, resetting the remaining actions to the value set here.

Definition at line 288 of file hs_platform_cfg.h.

Referenced by HS_AppInit().

10.4.2.24 HS_MISSION_REV

```
#define HS_MISSION_REV 0
```

Mission specific version number for HS application.

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 and the other parts are defined in "hs_version.h".

Limits:

Must be defined as a numeric value that is greater than or equal to zero.

Definition at line 657 of file hs_platform_cfg.h.

Referenced by HS_AppInit(), and HS_NoopCmd().

10.4.2.25 HS_POST_PROCESSING_DELAY

```
#define HS_POST_PROCESSING_DELAY 0
```

Time to wait after performing processing (in milliseconds)

Description:

Dictates the length of a task delay performed prior to checking the Software Bus for a Wakeup Message. This ensures that HS will run no more often than a certain rate. If this parameter is set to 0, no task delay will be performed. Time is in milliseconds.

Limits

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 223 of file hs_platform_cfg.h.

Referenced by HS_AppMain().

10.4.2.26 HS_RESET_TASK_DELAY

```
#define HS_RESET_TASK_DELAY 50
```

Time to wait before a processor reset (in milliseconds)

Description:

Dictates the length of the task delay (milliseconds) performed prior to calling [CFE_ES_ResetCFE](#) to allow for any event message to go out.

Limits

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 339 of file hs_platform_cfg.h.

Referenced by HS_MonitorApplications(), and HS_MonitorEvent().

10.4.2.27 HS_STARTUP_SYNC_TIMEOUT

```
#define HS_STARTUP_SYNC_TIMEOUT 65000
```

Time to wait for all apps to be started (in milliseconds)

Description:

Dictates the timeout for the [CFE_ES_WaitForStartupSync](#) call that HS uses to wait for all of the Applications specified in the startup script to finish initialization. HS will wait this amount of time before assuming all startup script applications have been started and will then begin nominal processing.

Limits

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This should be greater than or equal to the Startup Sync timeout for any application in the Application Monitor Table.

Definition at line 358 of file hs_platform_cfg.h.

Referenced by HS_AppMain().

10.4.2.28 HS_UTIL_AVERAGE_NUM_INTERVAL

```
#define HS_UTIL_AVERAGE_NUM_INTERVAL 4
```

CPU Average Utilization Number of Intervals.

Description:

Number of intervals over which the average utilization is computed.

Limits:

This parameter can't be larger than [HS_UTIL_PEAK_NUM_INTERVAL](#).

Definition at line 609 of file hs_platform_cfg.h.

Referenced by HS_MonitorUtilization().

10.4.2.29 HS_UTIL_CALLS_PER_MARK

```
#define HS_UTIL_CALLS_PER_MARK 1
```

CPU Utilization Calls per Mark (custom)

Description:

Number of times the Mark function must be called before it actually marks the time. This influences the interval size. The function calling the Mark function may not run at the same rate as the HS cycle (or HS may not want to monitor utilization every cycle) so this the interval to be at least as long as an HS cycle. Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 476 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.30 HS_UTIL_CONV_DIV

```
#define HS_UTIL_CONV_DIV 10000
```

CPU Utilization Conversion Factor Division (custom)

Description:

Division conversion factor. Number of idle ticks is divided by this value after it has been multiplied by [HS_UTIL_CONV_MULT1](#). Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

There may be processor dependent limits on value.

The result of the conversion must be a 32 bit signed integer (between -2147483648 and 2147483647).

Definition at line 552 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.31 HS_UTIL_CONV_MULT1

```
#define HS_UTIL_CONV_MULT1 1
```

CPU Utilization Conversion Factor Multiplication 1 (custom)

Description:

First multiplication conversion factor. Number of idle ticks is multiplied this value first when converting to utils. Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

There may be processor dependent limits on value. Note that all math is done using "uint32" values; it is important that the number of loop iterations in HS_IDLE, times this value, not overflow.

The result of the conversion must be a 32 bit signed integer (between -2147483648 and 2147483647).

Definition at line 535 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.32 HS_UTIL_CONV_MULT2

```
#define HS_UTIL_CONV_MULT2 1
```

CPU Utilization Conversion Factor Multiplication 2 (custom)

Description:

Second multiplication conversion factor. Number of idle ticks is multiplied this value after being divided by [HS_UTIL_CONV_DIV](#) after being multiplied by [HS_UTIL_CONV_MULT1](#) when converting to utils. Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

There may be processor dependent limits on value.

The result of the conversion must be a 32 bit signed integer (between -2147483648 and 2147483647).

Definition at line 570 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.33 HS_UTIL_CYCLES_PER_INTERVAL

```
#define HS_UTIL_CYCLES_PER_INTERVAL 1
```

CPU Utilization Cycles per Interval (custom)

Description:

Number of HS Cycles it takes to complete a CPU Utilization Interval. HS will monitor the utilization after this number of HS wakeup cycles. Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 491 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomMonitorUtilization\(\)](#).

10.4.2.34 HS_UTIL_DIAG_MASK

```
#define HS_UTIL_DIAG_MASK 0xFFFFFFFF
```

CPU Utilization Diagnostics Mask (custom)

Description:

Count mask for CPU Utilization Calibration. Time will be marked when (Counts & Mask) == Mask Note that these values are only necessarily relevant in the default [hs_custom.c](#).

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 624 of file [hs_platform_cfg.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.4.2.35 HS_UTIL_HOGGING_TIMEOUT

```
#define HS_UTIL_HOGGING_TIMEOUT 5
```

CPU Utilization Hogging Timeout.

Description:

Number of intervals for which the hogging limit must be exceeded before hogging is reported.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 583 of file [hs_platform_cfg.h](#).

Referenced by [HS_AppInit\(\)](#).

10.4.2.36 HS_UTIL_PEAK_NUM_INTERVAL

```
#define HS_UTIL_PEAK_NUM_INTERVAL 64
```

CPU Peak Utilization Number of Intervals.

Description:

Number of intervals over which the peak utilization is determined.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This parameter controls the size of the array which stores previously measured utilization values.

Definition at line 598 of file hs_platform_cfg.h.

Referenced by HS_MonitorUtilization().

10.4.2.37 HS_UTIL_PER_INTERVAL_HOGGING

```
#define HS_UTIL_PER_INTERVAL_HOGGING 9900
```

CPU Utilization Hogging Utils Per Interval.

Description:

Number of Utils (counts) equal to utilization which is considered hogging during one interval. A greater number of counts is also considered hogging.

Limits:

This parameter can't be larger than [HS_UTIL_PER_INTERVAL_TOTAL](#).

Definition at line 516 of file hs_platform_cfg.h.

Referenced by HS_MonitorUtilization().

10.4.2.38 HS_UTIL_PER_INTERVAL_TOTAL

```
#define HS_UTIL_PER_INTERVAL_TOTAL 10000
```

CPU Utilization Total Utils Per Interval.

Description:

Number of Utils (counts) equal to full utilization. This allows for higher resolution than percentages, and non decimal based values.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

Definition at line 504 of file `hs_platform_cfg.h`.

Referenced by `HS_CustomGetUtil()`, and `HS_MonitorUtilization()`.

10.4.2.39 HS_UTIL_TIME_DIAG_ARRAY_LENGTH

```
#define HS_UTIL_TIME_DIAG_ARRAY_LENGTH (1 << (HS_UTIL_TIME_DIAG_ARRAY_POWER))
```

Definition at line 640 of file `hs_platform_cfg.h`.

Referenced by `HS_UtilDiagReport()`.

10.4.2.40 HS_UTIL_TIME_DIAG_ARRAY_MASK

```
#define HS_UTIL_TIME_DIAG_ARRAY_MASK (HS_UTIL_TIME_DIAG_ARRAY_LENGTH - 1)
```

Definition at line 641 of file `hs_platform_cfg.h`.

Referenced by `HS_CustomInit()`.

10.4.2.41 HS_UTIL_TIME_DIAG_ARRAY_POWER

```
#define HS_UTIL_TIME_DIAG_ARRAY_POWER 4
```

CPU Utilization Diagnostics Array Configuration (custom)

Description:

Time will be marked into an array of subseconds. The independant parameter controls the exponent to which 2 is raised to determine the array size. As such, large values will require significant memory usage. Note that these values are only necessarily relevant in the default `hs_custom.c`.

Limits:

This parameter must be less than 32 and may not be negative.

Definition at line 639 of file `hs_platform_cfg.h`.

10.4.2.42 HS_WAKEUP_PIPE_DEPTH

```
#define HS_WAKEUP_PIPE_DEPTH 1
```

Software bus wakeup pipe depth.

Description:

Depth of the software bus pipe HS uses for wakeup messages. Used during initialization in the call to [CFE_SB_CreatePipe](#)

Limits:

This parameter must be greater than 0.

Definition at line 326 of file hs_platform_cfg.h.

Referenced by HS_SblInit().

10.4.2.43 HS_WAKEUP_TIMEOUT

```
#define HS_WAKEUP_TIMEOUT 1200
```

Wakeup Message Software Bus Timeout.

Description:

This parameter is passed into [CFE_SB_ReceiveBuffer](#) as the timeout value. It can specify [CFE_SB_POLL](#), [CFE_SB_PEND_FOREVER](#), or a timeout value in milliseconds.

Limits

This Parameter must be [CFE_SB_POLL](#), [CFE_SB_PEND_FOREVER](#), or greater than 0 and less than $2^{31} - 1$

As a timeout, this parameter should be less than ([HS_WATCHDOG_TIMEOUT_VALUE](#) * 1000) - HS runtime in ms otherwise HS may not be able to service the watchdog in time.

Definition at line 241 of file hs_platform_cfg.h.

Referenced by HS_AppMain().

10.4.2.44 HS_WATCHDOG_TIMEOUT_VALUE

```
#define HS_WATCHDOG_TIMEOUT_VALUE 10000
```

Watchdog Timeout Value.

Description:

Number of milliseconds before a watchdog timeout occurs.

Limits:

This parameter can't be larger than an unsigned 32 bit integer (4294967295).

This parameter must be greater than 0.

Definition at line 208 of file hs_platform_cfg.h.

Referenced by HS_AppMain().

10.4.2.45 HS_XCT_FILENAME

```
#define HS_XCT_FILENAME "/cf/hs_xct.tbl"
```

Execution Counter Table (XCT) filename.

Description:

Default file to load the Execution Counters Table from during a power-on reset sequence

Limits:

This string shouldn't be longer than [OS_MAX_PATH_LEN](#) for the target platform in question

Definition at line 445 of file hs_platform_cfg.h.

Referenced by HS_TblInit().

10.5 CFS Health and Safety Event IDs

Macros

- #define HS_INIT_EID 1
HS Initialization Event ID.
- #define HS_APP_EXIT_EID 2
HS Application Fatal Termination Event ID.
- #define HS_CDS_RESTORE_ERR_EID 3
HS CDS Restore Failed Event ID.
- #define HS_CR_CMD_PIPE_ERR_EID 4
HS Create Software Bus Command Pipe Failed Event ID.
- #define HS_CR_EVENT_PIPE_ERR_EID 5
HS Create Software Bus Event Pipe Failed Event ID.
- #define HS_CR_WAKEUP_PIPE_ERR_EID 6
HS Create Software Bus Wakeup Pipe Failed Event ID.
- #define HS_SUB_LONG_EVS_ERR_EID 7
HS Initialization Subscribe To Long Events Failed Event ID.
- #define HS_SUB_REQ_ERR_EID 8
HS Subscribe To Housekeeping Request Failed Event ID.
- #define HS_SUB_CMD_ERR_EID 9
HS Subscribe To Ground Commands Failed Event ID.
- #define HS_AMT_REG_ERR_EID 10
HS AppMon Table Register Failed Event ID.
- #define HS_EMT_REG_ERR_EID 11
HS EventMon Table Register Failed Event ID.
- #define HS_XCT_REG_ERR_EID 12
HS ExeCount Table Register Failed Event ID.
- #define HS_MAT_REG_ERR_EID 13
HS MsgActs Table Register Failed Event ID.
- #define HS_AMT_LD_ERR_EID 14
HS AppMon Table Load Failed Event ID.
- #define HS_EMT_LD_ERR_EID 15
HS EventMon Table Load Failed Event ID.
- #define HS_XCT_LD_ERR_EID 16
HS ExeCount Table Load Failed Event ID.
- #define HS_MAT_LD_ERR_EID 17
HS MsgActs Table Load Failed Event ID.
- #define HS_CDS_CORRUPT_ERR_EID 18
HS CDS Data Corrupt Resetting Data Event ID.
- #define HS_CC_ERR_EID 19
HS Command Code Invalid Event ID.
- #define HS_MID_ERR_EID 20
HS Command Pipe Message ID Invalid Event ID.
- #define HS_HKREQ_LEN_ERR_EID 21
HS Housekeeping Request Message Length Invalid Event ID.
- #define HS_LEN_ERR_EID 22

- `#define HS_NOOP_INF_EID 23`
HS Ground Command Length Invalid Event ID.
- `#define HS_RESET_DBG_EID 24`
HS No-op Command Event ID.
- `#define HS_ENABLE_APPMON_DBG_EID 25`
HS Reset Counters Command Event ID.
- `#define HS_DISABLE_APPMON_DBG_EID 26`
HS Enable Application Monitoring Command Event ID.
- `#define HS_ENABLE_EVENTMON_DBG_EID 27`
HS Disable Application Monitoring Command Event ID.
- `#define HS_DISABLE_EVENTMON_DBG_EID 28`
HS Enable Event Monitoring Command Event ID.
- `#define HS_ENABLE_ALIVENESS_DBG_EID 29`
HS Disable Event Monitoring Command Event ID.
- `#define HS_DISABLE_ALIVENESS_DBG_EID 30`
HS Enable Aliveness Indicator Command Event ID.
- `#define HS_RESET_RESETS_DBG_EID 31`
HS Disable Aliveness Indicator Command Event ID.
- `#define HS_SET_MAX_RESETS_DBG_EID 32`
HS Reset Process Reset Counter Command Event ID.
- `#define HS_APPMON_GETADDR_ERR_EID 33`
HS Set Max Resets Command Event ID.
- `#define HS_EVENTMON_GETADDR_ERR_EID 34`
HS AppMon Table Get Address Failed Event ID.
- `#define HS_EXECOUNT_GETADDR_ERR_EID 35`
HS EventMon Table Get Address Failed Event ID.
- `#define HS_MSGACTS_GETADDR_ERR_EID 36`
HS ExeCount Table Get Address Failed Event ID.
- `#define HS_RESET_LIMIT_ERR_EID 37`
HS MsgActs Table Get Address Failed Event ID.
- `#define HS_APPMON_APPNAME_ERR_EID 38`
HS Processor Reset Action Limit Reached No Reset Performed Event ID.
- `#define HS_APPMON_RESTART_ERR_EID 39`
HS App Monitor App Name Not Found Event ID.
- `#define HS_APPMON_NOT_RESTARTED_ERR_EID 40`
HS App Monitor Application Restart Event ID.
- `#define HS_APPMON_FAIL_ERR_EID 41`
HS App Monitor Application Restart Failed Event ID.
- `#define HS_APPMON_PROC_ERR_EID 42`
HS App Monitor Event Only Event ID.
- `#define HS_APPMON_MSGACTS_ERR_EID 43`
HS App Monitor Processor Reset Event ID.
- `#define HS_EVENTMON_MSGACTS_ERR_EID 44`
HS App Monitor Message Action Event ID.
- `#define HS_EVENTMON_PROC_ERR_EID 45`
HS Event Monitor Message Action Event ID.
- `#define HS_EVENTMON_RESET_ERR_EID 46`
HS Event Monitor Processor Reset Event ID.

- #define HS_EVENTMON_RESTART_ERR_EID 46
HS Event Monitor Application Restart Event ID.
- #define HS_EVENTMON_NOT_RESTARTED_ERR_EID 47
HS Event Monitor Application Restart Failed Event ID.
- #define HS_EVENTMON_DELETE_ERR_EID 48
HS Event Monitor Application Delete Event ID.
- #define HS_EVENTMON_NOT_DELETED_ERR_EID 49
HS Event Monitor Application Delete Failed Event ID.
- #define HS_AMTVAL_INF_EID 50
HS AppMon Table Verification Results Event ID.
- #define HS_AMTVAL_ERR_EID 51
HS AppMon Table Verification Failed Results Event ID.
- #define HS_EMTVAL_INF_EID 52
HS EventMon Table Verification Results Event ID.
- #define HS_EMTVAL_ERR_EID 53
HS EventMon Table Verification Failed Event ID.
- #define HS_XCTVAL_INF_EID 54
HS ExeCount Table Verification Results Event ID.
- #define HS_XCTVAL_ERR_EID 55
HS ExeCount Table Verification Failed Event ID.
- #define HS_MATVAL_INF_EID 56
HS MsgActs Table Verification Results Event ID.
- #define HS_MATVAL_ERR_EID 57
HS MsgActs Table Verification Failed Event ID.
- #define HS_DISABLE_APPMON_ERR_EID 58
HS Application Monitoring Disabled Due To Table Load Failure Event ID.
- #define HS_DISABLE_EVENTMON_ERR_EID 59
HS Event Monitoring Disabled Due To Table Load Failure Event ID.
- #define HS_SUB_WAKEUP_ERR_EID 60
HS Subscribe To Wakeup Message Failed Event ID.
- #define HS_CPUMON_HOGGING_ERR_EID 61
HS CPU Hogging Detected Event ID.
- #define HS_ENABLE_CPUHOG_DBG_EID 64
HS CPU Hogging Detection Enabled Event ID.
- #define HS_DISABLE_CPUHOG_DBG_EID 65
HS CPU Hogging Detection Disabled Event ID.
- #define HS_EVENTMON_LONG_SUB_EID 66
HS Event Monitor Enable Subscribe To Long Events Failed Event ID.
- #define HS_EVENTMON_SHORT_SUB_EID 67
HS Event Monitor Enable Subscribe to Short Events Failed Event ID.
- #define HS_EVENTMON_LONG_UNSUB_EID 68
HS Event Monitor Disable Unsubscribe From Long Events Failed Event ID.
- #define HS_EVENTMON_SHORT_UNSUB_EID 69
HS Event Monitor Disable Unsubscribe From Short Events Failed Event ID.
- #define HS_BADEMT_LONG_UNSUB_EID 70
HS EventMon Table Bad Unsubscribing From Long Events Failed Event ID.
- #define HS_BADEMT_SHORT_UNSUB_EID 71

- #define HS_APPMON_APPNAME_DBG_EID 72
HS App Monitor App Name Not Found Event ID.
 - #define HS_HKREQ_RESOURCE_DBG_EID 73
HS Housekeeping Request Unknown Resource Event ID.
 - #define HS_CUSTOM_INIT_ERR_EID 74
HS Custom Initialization Failed Event ID.
 - #define HS_AM_TBL_NULL_ERR_EID 75
HS AppMon Table Validation Null Pointer Event ID.
 - #define HS_EM_TBL_NULL_ERR_EID 76
HS EventMon Table Validation Null Pointer Event ID.
 - #define HS_XC_TBL_NULL_ERR_EID 77
HS ExeCount Table Validation Null Pointer Event ID.
 - #define HS_MA_TBL_NULL_ERR_EID 78
HS MsgActs Table Validation Null Pointer Event ID.
 - #define HS_SUB_SHORT_EVTS_ERR_EID 79
HS Initialization Subscribe To Short Events Failed Event ID.

 - #define HS_CR_CHILD_TASK_ERR_EID 101
HS CPU Utilization Monitoring Create Child Task Failed Event ID.
 - #define HS_CR_SYNC_CALLBACK_ERR_EID 102
HS CPU Utilization Monitoring Register Sync Callback Failed Event ID.
 - #define HS_UTIL_DIAG_REPORT_EID 103
HS Report Diagnostics Command Event ID.
 - #define HS_SET_UTIL_PARAMS_DBG_EID 104
HS Set Utilization Paramaters Command Event ID.
 - #define HS_SET_UTIL_PARAMS_ERR_EID 105
HS Set Utilization Parameters Zero Value Event ID.
 - #define HS_SET_UTIL_DIAG_DBG_EID 106
HS Set Utilization Diagnostics Command Event ID.

10.5.1 Detailed Description

10.5.2 Macro Definition Documentation

10.5.2.1 HS_AM_TBL_NULL_ERR_EID

```
#define HS_AM_TBL_NULL_ERR_EID 75
```

HS AppMon Table Validation Null Pointer Event ID.

Type: ERROR

Cause:

This event message is issued if the TableData pointer passed to HS_ValidateAMTable is null.

Definition at line 928 of file hs_events.h.

Referenced by HS_ValidateAMTable().

10.5.2.2 HS_AMT_LD_ERR_EID

```
#define HS_AMT_LD_ERR_EID 14
```

HS AppMon Table Load Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to CFE_TBL_Load for the application monitor table returns a value other than CFE_SUCCESS

Definition at line 204 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.3 HS_AMT_REG_ERR_EID

```
#define HS_AMT_REG_ERR_EID 10
```

HS AppMon Table Register Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when Health and Safety is unable to register its Application Monitor Table with cFE Table Services via the [CFE_TBL_Register](#) API.

Definition at line 156 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.4 HS_AMTVAL_ERR_EID

```
#define HS_AMTVAL_ERR_EID 51
```

HS AppMon Table Verification Failed Results Event ID.

Type: **ERROR**

Cause:

This event message is issued on the first error when a table validation fails for a application monitor table load.

Definition at line 653 of file hs_events.h.

Referenced by HS_ValidateAMTable().

10.5.2.5 HS_AMTVAL_INF_EID

```
#define HS_AMTVAL_INF_EID 50
```

HS AppMon Table Verification Results Event ID.

Type: INFORMATIONAL

Cause:

This event message is issued when a table validation has been completed for an application monitor table load

Definition at line 641 of file hs_events.h.

Referenced by HS_ValidateAMTable().

10.5.2.6 HS_APP_EXIT_EID

```
#define HS_APP_EXIT_EID 2
```

HS Application Fatal Termination Event ID.

Type: CRITICAL

Cause:

This event message is issued when CFS Health and Safety exits due to a fatal error condition.

Definition at line 54 of file hs_events.h.

Referenced by HS_AppMain().

10.5.2.7 HS_APPMON_APPNAME_DBG_EID

```
#define HS_APPMON_APPNAME_DBG_EID 72
```

HS App Monitor App Name Not Found Event ID.

Type: DEBUG

Cause:

This event message is issued when an application name cannot be resolved into an application ID by the OS. This event is sent if this error occurs repeatedly in the HS_MonitorApplications function. The first such occurrence is captured with a corresponding error event.

Definition at line 891 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.8 HS_APPMON_APPNAME_ERR_EID

```
#define HS_APPMON_APPNAME_ERR_EID 38
```

HS App Monitor App Name Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when an application name cannot be resolved into an application ID by the OS. This event is sent the first time this error occurs in the HS_MonitorApplications function. Subsequent occurrences are captured with a corresponding debug event.

Definition at line 493 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.9 HS_APPMON_FAIL_ERR_EID

```
#define HS_APPMON_FAIL_ERR_EID 41
```

HS App Monitor Event Only Event ID.

Type: **ERROR**

Cause:

This event message is issued when a monitored application fails to increment its execution counter in the table specified number of cycles, and the specified action type is Event Only.

Definition at line 531 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.10 HS_APPMON_GETADDR_ERR_EID

```
#define HS_APPMON_GETADDR_ERR_EID 33
```

HS AppMon Table Get Address Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the address can't be obtained from table services for the applications monitor table.

Definition at line 431 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.11 HS_APPMON_MSGACTS_ERR_EID

```
#define HS_APPMON_MSGACTS_ERR_EID 43
```

HS App Monitor Message Action Event ID.

Type: **ERROR**

Cause:

This event message is issued when a monitored application fails to increment its execution counter in the table specified number of cycles, and the specified action type is a Message Action.

Definition at line 557 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.12 HS_APPMON_NOT_RESTARTED_ERR_EID

```
#define HS_APPMON_NOT_RESTARTED_ERR_EID 40
```

HS App Monitor Application Restart Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the app monitor attempts to restart a task and is unable to.

Definition at line 518 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.13 HS_APPMON_PROC_ERR_EID

```
#define HS_APPMON_PROC_ERR_EID 42
```

HS App Monitor Processor Reset Event ID.

Type: **ERROR**

Cause:

This event message is issued when a monitored application fails to increment its execution counter in the table specified number of cycles, and the specified action type is Processor Reset.

Definition at line 544 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.14 HS_APPMON_RESTART_ERR_EID

```
#define HS_APPMON_RESTART_ERR_EID 39
```

HS App Monitor Application Restart Event ID.

Type: **ERROR**

Cause:

This event message is issued when a monitored application fails to increment its execution counter in the table specified number of cycles, and the specified action type is Application Restart.

Definition at line 506 of file hs_events.h.

Referenced by HS_MonitorApplications().

10.5.2.15 HS_BADEMT_LONG_UNSUB_EID

```
#define HS_BADEMT_LONG_UNSUB_EID 70
```

HS EventMon Table Bad Unsubscribing From Long Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued if when acquiring the event monitor table from table services, it is bad and event monitoring is disabled, but there is a failure unsubscribing from the event mid.

Definition at line 864 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.16 HS_BADEMT_SHORT_UNSUB_EID

```
#define HS_BADEMT_SHORT_UNSUB_EID 71
```

HS EventMon Table Bad Unsubscribing From Short Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued if when acquiring the event monitor table from table services, it is bad and event monitoring is disabled, but there is a failure unsubscribing from the event mid.

Definition at line 877 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.17 HS_CC_ERR_EID

```
#define HS_CC_ERR_EID 19
```

HS Command Code Invalid Event ID.

Type: **ERROR**

Cause:

This event message is issued when a software bus message is received with an invalid command code.

Definition at line 264 of file hs_events.h.

Referenced by HS_AppPipe().

10.5.2.18 HS_CDS_CORRUPT_ERR_EID

```
#define HS_CDS_CORRUPT_ERR_EID 18
```

HS CDS Data Corrupt Resetting Data Event ID.

Type: **ERROR**

Cause:

This event message is issued when CFS Health and Safety restores data from the CDS that does not pass validation

Definition at line 252 of file hs_events.h.

Referenced by HS_AppInit().

10.5.2.19 HS_CDS_RESTORE_ERR_EID

```
#define HS_CDS_RESTORE_ERR_EID 3
```

HS CDS Restore Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFS Health and Safety is unable to restore data from its critical data store

Definition at line 66 of file hs_events.h.

Referenced by HS_AppInit().

10.5.2.20 HS_CPUMON_HOGGING_ERR_EID

```
#define HS_CPUMON_HOGGING_ERR_EID 61
```

HS CPU Hogging Detected Event ID.

Type: ERROR

Cause:

This event message is issued when the CPU monitoring detects that the utilization has exceeded the CPU Hogging threshhold for longer than the CPU Hogging duration

Definition at line 775 of file hs_events.h.

Referenced by HS_MonitorUtilization().

10.5.2.21 HS_CR_CHILD_TASK_ERR_EID

```
#define HS_CR_CHILD_TASK_ERR_EID 101
```

HS CPU Utilization Monitoring Create Child Task Failed Event ID.

Custom Event IDs must not conflict with those in [hs_events.h](#)

Type: **ERROR**

Cause:

This event message is issued when CFS Health and Safety is unable to create its child task via the [CFE_ES_CreateChildTask](#) API

Definition at line 146 of file [hs_custom.h](#).

Referenced by [HS_CustomInit\(\)](#).

10.5.2.22 HS_CR_CMD_PIPE_ERR_EID

```
#define HS_CR_CMD_PIPE_ERR_EID 4
```

HS Create Software Bus Command Pipe Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when CFS Health and Safety is unable to create its command pipe via the [CFE_SB_CreatePipe](#) API

Definition at line 79 of file [hs_events.h](#).

Referenced by [HS_SbInit\(\)](#).

10.5.2.23 HS_CR_EVENT_PIPE_ERR_EID

```
#define HS_CR_EVENT_PIPE_ERR_EID 5
```

HS Create Software Bus Event Pipe Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when CFS Health and Safety is unable to create its event pipe via the [CFE_SB_CreatePipe API](#)

Definition at line 92 of file hs_events.h.

Referenced by HS_SbInit().

10.5.2.24 HS_CR_SYNC_CALLBACK_ERR_EID

```
#define HS_CR_SYNC_CALLBACK_ERR_EID 102
```

HS CPU Utilization Monitoring Register Sync Callback Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when CFS Health and Safety is unable to create its sync callback via the [CFE_TIME_RegisterSyncCallback API](#)

Definition at line 159 of file hs_custom.h.

Referenced by HS_CustomInit().

10.5.2.25 HS_CR_WAKEUP_PIPE_ERR_EID

```
#define HS_CR_WAKEUP_PIPE_ERR_EID 6
```

HS Create Software Bus Wakeup Pipe Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFS Health and Safety is unable to create its wakeup pipe via the [CFE_SB_CreatePipe API](#)

Definition at line 105 of file hs_events.h.

Referenced by HS_SbInit().

10.5.2.26 HS_CUSTOM_INIT_ERR_EID

```
#define HS_CUSTOM_INIT_ERR_EID 74
```

HS Custom Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued if the HS_CustomInit routine returns any value other than CFE_SUCCESS.

Definition at line 916 of file hs_events.h.

Referenced by HS_AppInit().

10.5.2.27 HS_DISABLE_ALIVENESS_DBG_EID

```
#define HS_DISABLE_ALIVENESS_DBG_EID 30
```

HS Disable Aliveness Indicator Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a disable aliveness indicator command has been received.

Definition at line 395 of file hs_events.h.

Referenced by HS_DisableAlivenessCmd().

10.5.2.28 HS_DISABLE_APPMON_DBG_EID

```
#define HS_DISABLE_APPMON_DBG_EID 26
```

HS Disable Application Monitoring Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a disable application monitoring command has been received.

Definition at line 347 of file hs_events.h.

Referenced by HS_DisableAppMonCmd().

10.5.2.29 HS_DISABLE_APPMON_ERR_EID

```
#define HS_DISABLE_APPMON_ERR_EID 58
```

HS Application Monitoring Disabled Due To Table Load Failure Event ID.

Type: **ERROR**

Cause:

This event message is issued when application monitoring has been disabled due to a table load failure.

Definition at line 737 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.30 HS_DISABLE_CPUHOG_DBG_EID

```
#define HS_DISABLE_CPUHOG_DBG_EID 65
```

HS CPU Hogging Detection Disabled Event ID.

Type: **DEBUG**

Cause:

This event message is issued when a disable cpu hogging indicator command has been received.

Definition at line 799 of file hs_events.h.

Referenced by HS_DisableCPUHogCmd().

10.5.2.31 HS_DISABLE_EVENTMON_DBG_EID

```
#define HS_DISABLE_EVENTMON_DBG_EID 28
```

HS Disable Event Monitoring Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a disable event monitoring command has been received.

Definition at line 371 of file hs_events.h.

Referenced by HS_DisableEventMonCmd().

10.5.2.32 HS_DISABLE_EVENTMON_ERR_EID

```
#define HS_DISABLE_EVENTMON_ERR_EID 59
```

HS Event Monitoring Disabled Due To Table Load Failure Event ID.

Type: ERROR

Cause:

This event message is issued when event monitoring has been disabled due to a table load failure.

Definition at line 749 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.33 HS_EM_TBL_NULL_ERR_EID

```
#define HS_EM_TBL_NULL_ERR_EID 76
```

HS EventMon Table Validation Null Pointer Event ID.

Type: ERROR

Cause:

This event message is issued if the TableData pointer passed to HS_ValidateEMTable is null.

Definition at line 940 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.5.2.34 HS_EMT_LD_ERR_EID

```
#define HS_EMT_LD_ERR_EID 15
```

HS EventMon Table Load Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to CFE_TBL_Load for the event monitor table returns a value other than CFE_SUCCESS

Definition at line 216 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.35 HS_EMT_REG_ERR_EID

```
#define HS_EMT_REG_ERR_EID 11
```

HS EventMon Table Register Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when Health and Safety is unable to register its Event Monitor Table with cFE Table Services via the [CFE_TBL_Register](#) API.

Definition at line 168 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.36 HS_EMTVAL_ERR_EID

```
#define HS_EMTVAL_ERR_EID 53
```

HS EventMon Table Verification Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued on the first error when a table validation fails for an event monitor table load.

Definition at line 677 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.5.2.37 HS_EMTVAL_INF_EID

```
#define HS_EMTVAL_INF_EID 52
```

HS EventMon Table Verification Results Event ID.

Type: INFORMATIONAL

Cause:

This event message is issued when a table validation has been completed for an event monitor table load

Definition at line 665 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.5.2.38 HS_ENABLE_ALIVENESS_DBG_EID

```
#define HS_ENABLE_ALIVENESS_DBG_EID 29
```

HS Enable Aliveness Indicator Command Event ID.

Type: DEBUG

Cause:

This event message is issued when an enable aliveness indicator command has been received.

Definition at line 383 of file hs_events.h.

Referenced by HS_EnableAlivenessCmd().

10.5.2.39 HS_ENABLE_APPMON_DBG_EID

```
#define HS_ENABLE_APPMON_DBG_EID 25
```

HS Enable Application Monitoring Command Event ID.

Type: DEBUG

Cause:

This event message is issued when an enable application monitoring command has been received.

Definition at line 335 of file hs_events.h.

Referenced by HS_EnableAppMonCmd().

10.5.2.40 HS_ENABLE_CPUHOG_DBG_EID

```
#define HS_ENABLE_CPUHOG_DBG_EID 64
```

HS CPU Hogging Detection Enabled Event ID.

Type: DEBUG

Cause:

This event message is issued when an enable cpu hogging indicator command has been received.

Definition at line 787 of file hs_events.h.

Referenced by HS_EnableCPUHogCmd().

10.5.2.41 HS_ENABLE_EVENTMON_DBG_EID

```
#define HS_ENABLE_EVENTMON_DBG_EID 27
```

HS Enable Event Monitoring Command Event ID.

Type: DEBUG

Cause:

This event message is issued when an enable event monitoring command has been received.

Definition at line 359 of file hs_events.h.

Referenced by HS_EnableEventMonCmd().

10.5.2.42 HS_EVENTMON_DELETE_ERR_EID

```
#define HS_EVENTMON_DELETE_ERR_EID 48
```

HS Event Monitor Application Delete Event ID.

Type: ERROR

Cause:

This event message is issued when an event is received that matches an event in the event monitor table that specifies Delete Application as the action type

Definition at line 617 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.43 HS_EVENTMON_GETADDR_ERR_EID

```
#define HS_EVENTMON_GETADDR_ERR_EID 34
```

HS EventMon Table Get Address Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the address can't be obtained from table services for the event monitor table.

Definition at line 443 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.44 HS_EVENTMON_LONG_SUB_EID

```
#define HS_EVENTMON_LONG_SUB_EID 66
```

HS Event Monitor Enable Subscribe To Long Events Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when a ground command message is received to enable event monitoring while it is disabled, and there is an error subscribing to the long-format event mid ([CFE_EVS_LONG_EVENT_MSG_MID](#)).

Definition at line 812 of file hs_events.h.

Referenced by HS_EnableEventMonCmd().

10.5.2.45 HS_EVENTMON_LONG_UNSUB_EID

```
#define HS_EVENTMON_LONG_UNSUB_EID 68
```

HS Event Monitor Disable Unsubscribe From Long Events Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when a ground command message is received to disable event monitoring while it is enabled, and there is an error unsubscribing from the long-format event mid ([CFE_EVS_LONG_EVENT_MSG_MID](#)).

Definition at line 838 of file hs_events.h.

Referenced by HS_DisableEventMonCmd().

10.5.2.46 HS_EVENTMON_MSGACTS_ERR_EID

```
#define HS_EVENTMON_MSGACTS_ERR_EID 44
```

HS Event Monitor Message Action Event ID.

Type: **ERROR**

Cause:

This event message is issued when a monitored event is detected, and the specified action type is a Message Action.

Definition at line 569 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.47 HS_EVENTMON_NOT_DELETED_ERR_EID

```
#define HS_EVENTMON_NOT_DELETED_ERR_EID 49
```

HS Event Monitor Application Delete Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the event monitor attempts to delete a task and is unable to.

Definition at line 629 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.48 HS_EVENTMON_NOT_RESTARTED_ERR_EID

```
#define HS_EVENTMON_NOT_RESTARTED_ERR_EID 47
```

HS Event Monitor Application Restart Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the event monitor attempts to restart a task and is unable to.

Definition at line 605 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.49 HS_EVENTMON_PROC_ERR_EID

```
#define HS_EVENTMON_PROC_ERR_EID 45
```

HS Event Monitor Processor Reset Event ID.

Type: **ERROR**

Cause:

This event message is issued when an event is received that matches an event in the event monitor table that specifies Processor Reset as the action type

Definition at line 581 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.50 HS_EVENTMON_RESTART_ERR_EID

```
#define HS_EVENTMON_RESTART_ERR_EID 46
```

HS Event Monitor Application Restart Event ID.

Type: **ERROR**

Cause:

This event message is issued when an event is received that matches an event in the event monitor table that specifies Restart Application as the action type

Definition at line 593 of file hs_events.h.

Referenced by HS_MonitorEvent().

10.5.2.51 HS_EVENTMON_SHORT_SUB_EID

```
#define HS_EVENTMON_SHORT_SUB_EID 67
```

HS Event Monitor Enable Subscribe to Short Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued when a ground command message is received to enable event monitoring while it is disabled, and there is an error subscribing to the short-format event mid ([CFE_EVS_SHORT_EVENT_MSG_MID](#)).

Definition at line 825 of file hs_events.h.

Referenced by HS_EnableEventMonCmd().

10.5.2.52 HS_EVENTMON_SHORT_UNSUB_EID

```
#define HS_EVENTMON_SHORT_UNSUB_EID 69
```

HS Event Monitor Disable Unsubscribe From Short Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued when a ground command message is received to disable event monitoring while it is enabled, and there is an error unsubscribing from the long-format event mid ([CFE_EVS_SHORT_EVENT_MSG_MID](#)).

Definition at line 851 of file hs_events.h.

Referenced by HS_DisableEventMonCmd().

10.5.2.53 HS_EXECOUNT_GETADDR_ERR_EID

```
#define HS_EXECOUNT_GETADDR_ERR_EID 35
```

HS ExeCount Table Get Address Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the address can't be obtained from table services for the execution counters table.

Definition at line 455 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.54 HS_HKREQ_LEN_ERR_EID

```
#define HS_HKREQ_LEN_ERR_EID 21
```

HS Housekeeping Request Message Length Invalid Event ID.

Type: **ERROR**

Cause:

This event message is issued when a housekeeping request is received with a message length that doesn't match the expected value.

Definition at line 288 of file hs_events.h.

Referenced by HS_VerifyMsgLength().

10.5.2.55 HS_HKREQ_RESOURCE_DBG_EID

```
#define HS_HKREQ_RESOURCE_DBG_EID 73
```

HS Housekeeping Request Unknown Resource Event ID.

Type: DEBUG

Cause:

This event message is issued when the application encounters a resource with an unknown type while populating the execution counters in the housekeeping telemetry packet.

Definition at line 904 of file hs_events.h.

Referenced by HS_HousekeepingReq().

10.5.2.56 HS_INIT_EID

```
#define HS_INIT_EID 1
```

HS Initialization Event ID.

Type: INFORMATIONAL

Cause:

This event message is issued when the CFS Health and Safety has completed initialization.

Definition at line 42 of file hs_events.h.

Referenced by HS_AppInit().

10.5.2.57 HS_LEN_ERR_EID

```
#define HS_LEN_ERR_EID 22
```

HS Ground Command Length Invalid Event ID.

Type: **ERROR**

Cause:

This event message is issued when a ground command message is received with a message length that doesn't match the expected value.

Definition at line 300 of file hs_events.h.

Referenced by HS_VerifyMsgLength().

10.5.2.58 HS_MA_TBL_NULL_ERR_EID

```
#define HS_MA_TBL_NULL_ERR_EID 78
```

HS MsgActs Table Validation Null Pointer Event ID.

Type: **ERROR**

Cause:

This event message is issued if the TableData pointer passed to HS_ValidateMATable is null.

Definition at line 964 of file hs_events.h.

Referenced by HS_ValidateMATable().

10.5.2.59 HS_MAT_LD_ERR_EID

```
#define HS_MAT_LD_ERR_EID 17
```

HS MsgActs Table Load Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the call to CFE_TBL_Load for the message actions table returns a value other than CFE_SUCCESS

Definition at line 240 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.60 HS_MAT_REG_ERR_EID

```
#define HS_MAT_REG_ERR_EID 13
```

HS MsgActs Table Register Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when Health and Safety is unable to register its Message Actions Table with cFE Table Services via the [CFE_TBL_Register](#) API.

Definition at line 192 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.61 HS_MATVAL_ERR_EID

```
#define HS_MATVAL_ERR_EID 57
```

HS MsgActs Table Verification Failed Event ID.

Type: ERROR

Cause:

This event message is issued on the first error when a table validation fails for an message actions table load.

Definition at line 725 of file hs_events.h.

Referenced by HS_ValidateMATable().

10.5.2.62 HS_MATVAL_INF_EID

```
#define HS_MATVAL_INF_EID 56
```

HS MsgActs Table Verification Results Event ID.

Type: INFORMATIONAL

Cause:

This event message is issued when a table validation has been completed for an message actions table load

Definition at line 713 of file hs_events.h.

Referenced by HS_ValidateMATable().

10.5.2.63 HS_MID_ERR_EID

```
#define HS_MID_ERR_EID 20
```

HS Command Pipe Message ID Invalid Event ID.

Type: ERROR

Cause:

This event message is issued when a software bus message is received with an invalid message ID.

Definition at line 276 of file hs_events.h.

Referenced by HS_AppPipe().

10.5.2.64 HS_MSGACTS_GETADDR_ERR_EID

```
#define HS_MSGACTS_GETADDR_ERR_EID 36
```

HS MsgActs Table Get Address Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the address can't be obtained from table services for the message actions table.

Definition at line 467 of file hs_events.h.

Referenced by HS_AcquirePointers().

10.5.2.65 HS_NOOP_INF_EID

```
#define HS_NOOP_INF_EID 23
```

HS No-op Command Event ID.

Type: INFORMATIONAL

Cause:

This event message is issued when a NOOP command has been received.

Definition at line 311 of file hs_events.h.

Referenced by HS_NoopCmd().

10.5.2.66 HS_RESET_DBG_EID

```
#define HS_RESET_DBG_EID 24
```

HS Reset Counters Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a reset counters command has been received.

Definition at line 323 of file hs_events.h.

Referenced by HS_ResetCmd().

10.5.2.67 HS_RESET_LIMIT_ERR_EID

```
#define HS_RESET_LIMIT_ERR_EID 37
```

HS Processor Reset Action Limit Reached No Reset Performed Event ID.

Type: ERROR

Cause:

This event message is issued when the action specified by an Application or Event monitor that fails is Processor Reset, and no more Processor Resets are allowed.

Definition at line 479 of file hs_events.h.

Referenced by HS_MonitorApplications(), and HS_MonitorEvent().

10.5.2.68 HS_RESET_RESETS_DBG_EID

```
#define HS_RESET_RESETS_DBG_EID 31
```

HS Reset Process Reset Counter Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a reset processor resets count command has been received.

Definition at line 407 of file hs_events.h.

Referenced by HS_ResetResetsPerformedCmd().

10.5.2.69 HS_SET_MAX_RESETS_DBG_EID

```
#define HS_SET_MAX_RESETS_DBG_EID 32
```

HS Set Max Resets Command Event ID.

Type: DEBUG

Cause:

This event message is issued when a set max processor resets command has been received.

Definition at line 419 of file hs_events.h.

Referenced by HS_SetMaxResetsCmd().

10.5.2.70 HS_SET_UTIL_DIAG_DBG_EID

```
#define HS_SET_UTIL_DIAG_DBG_EID 106
```

HS Set Utilization Diagnostics Command Event ID.

Type: Debug

Cause:

This event message is issued when CFS Health and Safety successfully processes the [HS_SET_UTIL_DIAG_CC](#) command.

Definition at line 204 of file hs_custom.h.

Referenced by HS_SetUtilDiagCmd().

10.5.2.71 HS_SET_UTIL_PARAMS_DBG_EID

```
#define HS_SET_UTIL_PARAMS_DBG_EID 104
```

HS Set Utilization Parameters Command Event ID.

Type: Debug

Cause:

This event message is issued when CFS Health and Safety successfully processes the [HS_SET_UTIL_PARAMS_CC](#) command.

Definition at line 181 of file hs_custom.h.

Referenced by HS_SetUtilParamsCmd().

10.5.2.72 HS_SET_UTIL_PARAMS_ERR_EID

```
#define HS_SET_UTIL_PARAMS_ERR_EID 105
```

HS Set Utilization Parameters Zero Value Event ID.

Type: Error

Cause:

This event message is issued when CFS Health and Safety fails to processes the [HS_SET_UTIL_PARAMS_CC](#) command. due to a 0 as at least one of the parameters.

Definition at line 193 of file hs_custom.h.

Referenced by HS_SetUtilParamsCmd().

10.5.2.73 HS_SUB_CMD_ERR_EID

```
#define HS_SUB_CMD_ERR_EID 9
```

HS Subscribe To Ground Commands Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to [CFE_SB_Subscribe](#) for the [HS_CMD_MID](#), during initialization returns a value other than CFE_SUCCESS

Definition at line 144 of file hs_events.h.

Referenced by HS_SbInit().

10.5.2.74 HS_SUB_LONG_EVTS_ERR_EID

```
#define HS_SUB_LONG_EVTS_ERR_EID 7
```

HS Initialization Subscribe To Long Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to [CFE_SB_Subscribe](#) for the [CFE_EVT_LONG_EVENT_MSG_MID](#), during initialization returns a value other than CFE_SUCCESS

Definition at line 118 of file hs_events.h.

Referenced by HS_AppMain().

10.5.2.75 HS_SUB_REQ_ERR_EID

```
#define HS_SUB_REQ_ERR_EID 8
```

HS Subscribe To Housekeeping Request Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to [CFE_SB_Subscribe](#) for the [HS_SEND_HK_MID](#), during initialization returns a value other than CFE_SUCCESS

Definition at line 131 of file hs_events.h.

Referenced by HS_SbInit().

10.5.2.76 HS_SUB_SHORT_EVT_ERR_EID

```
#define HS_SUB_SHORT_EVT_ERR_EID 79
```

HS Initialization Subscribe To Short Events Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the call to [CFE_SB_Subscribe](#) for the [CFE_EVT_SHORT_EVENT_MSG_MID](#), during initialization returns a value other than CFE_SUCCESS

Definition at line 977 of file hs_events.h.

Referenced by HS_AppMain().

10.5.2.77 HS_SUB_WAKEUP_ERR_EID

```
#define HS_SUB_WAKEUP_ERR_EID 60
```

HS Subscribe To Wakeup Message Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the call to [CFE_SB_Subscribe](#) for the [HS_WAKEUP_MID](#), during initialization returns a value other than [CFE_SUCCESS](#)

Definition at line 762 of file hs_events.h.

Referenced by [HS_SbInit\(\)](#).

10.5.2.78 HS_UTIL_DIAG_REPORT_EID

```
#define HS_UTIL_DIAG_REPORT_EID 103
```

HS Report Diagnostics Command Event ID.

Type: **INFORMATION**

Cause:

This event message is issued when CFS Health and Safety receives the [HS_REPORT_DIAG_CC](#) command.

Definition at line 170 of file hs_custom.h.

Referenced by [HS_UtilDiagReport\(\)](#).

10.5.2.79 HS_XC_TBL_NULL_ERR_EID

```
#define HS_XC_TBL_NULL_ERR_EID 77
```

HS ExeCount Table Validation Null Pointer Event ID.

Type: **ERROR**

Cause:

This event message is issued if the TableData pointer passed to HS_ValidateXCTable is null.

Definition at line 952 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.5.2.80 HS_XCT_LD_ERR_EID

```
#define HS_XCT_LD_ERR_EID 16
```

HS ExeCount Table Load Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when the call to CFE_TBL_Load for the execution counters table returns a value other than CFE_SUCCESS

Definition at line 228 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.81 HS_XCT_REG_ERR_EID

```
#define HS_XCT_REG_ERR_EID 12
```

HS ExeCount Table Register Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued when Health and Safety is unable to register its Execution Counters Table with cFE Table Services via the [CFE_TBL_Register](#) API.

Definition at line 180 of file hs_events.h.

Referenced by HS_TblInit().

10.5.2.82 HS_XCTVAL_ERR_EID

```
#define HS_XCTVAL_ERR_EID 55
```

HS ExeCount Table Verification Failed Event ID.

Type: **ERROR**

Cause:

This event message is issued on the first error when a table validation fails for an execution counter table load.

Definition at line 701 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.5.2.83 HS_XCTVAL_INF_EID

```
#define HS_XCTVAL_INF_EID 54
```

HS ExeCount Table Verification Results Event ID.

Type: **INFORMATIONAL**

Cause:

This event message is issued when a table validation has been completed for an execution counters table load

Definition at line 689 of file hs_events.h.

Referenced by HS_ValidateEMTable().

10.6 CFS Health and Safety Command Structures

Data Structures

- struct [HS_SetUtilParamsCmd_t](#)
Set Utililiztion Parameters Command.
- struct [HS_NoArgsCmd_t](#)
No Arguments Command.
- struct [HS_SetMaxResetsCmd_t](#)
Set Max Resets Command.

10.6.1 Detailed Description

10.7 CFS Health and Safety Telemetry

Data Structures

- struct [HS_HkPacket_t](#)
Housekeeping Packet Structure.

10.7.1 Detailed Description

10.8 CFS Health and Safety Command Codes

Macros

- `#define HS_NOOP_CC 0`
Noop.
- `#define HS_RESET_CC 1`
Reset Counters.
- `#define HS_ENABLE_APPMON_CC 2`
Enable Applications Monitor.
- `#define HS_DISABLE_APPMON_CC 3`
Disable Applications Monitor.
- `#define HS_ENABLE_EVENTMON_CC 4`
Enable Events Monitor.
- `#define HS_DISABLE_EVENTMON_CC 5`
Disable Events Monitor.
- `#define HS_ENABLE_ALIVENESS_CC 6`
Enable Aliveness Indicator.
- `#define HS_DISABLE_ALIVENESS_CC 7`
Disable Aliveness Indicator.
- `#define HS_RESET_RESETS_PERFORMED_CC 8`
Reset Processor Resets Performed Count.
- `#define HS_SET_MAX_RESETS_CC 9`
Set Max Processor Resets Performed Count.
- `#define HS_ENABLE_CPUHOG_CC 10`
Enable CPU Hogging Indicator.
- `#define HS_DISABLE_CPUHOG_CC 11`
Disable CPU Hogging Indicator.

- `#define HS_REPORT_DIAG_CC 12`
Report Util Diagnostics.
- `#define HS_SET_UTIL_PARAMS_CC 13`
Set Utilization Calibration Parameters.
- `#define HS_SET_UTIL_DIAG_CC 14`
Set Utilization Diagnostics Parameter.

10.8.1 Detailed Description

10.8.2 Macro Definition Documentation

10.8.2.1 HS_DISABLE_ALIVENESS_CC

```
#define HS_DISABLE_ALIVENESS_CC 7
```

Disable Aliveness Indicator.

Description

Disables the Aliveness Indicator UART output

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentAlivenessState](#) will be set to Disabled
- The [HS_DISABLE_ALIVENESS_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_ENABLE_ALIVENESS_CC](#)

Definition at line 323 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.2 HS_DISABLE_APPMON_CC

```
#define HS_DISABLE_APPMON_CC 3
```

Disable Applications Monitor.

Description

Disables the Applications Monitor

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentAppMonState](#) will be set to Disabled
- The [HS_DISABLE_APPMON_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_ENABLE_APPMON_CC](#)

Definition at line 195 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.3 HS_DISABLE_CPUHOG_CC

```
#define HS_DISABLE_CPUHOG_CC 11
```

Disable CPU Hogging Indicator.

Description

Disables the CPU Hogging Indicator Event Message

Command Structure

`HS_NoArgsCmd_t`

Command Verification

Successful execution of this command may be verified with the following telemetry:

- `HS_HkPacket_t.CmdCount` will increment
- `HS_HkPacket_t.CurrentCPUHogState` will be set to Disabled
- The `HS_DISABLE_CPUHOG_DBG_EID` informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- `HS_HkPacket_t.CmdErrCount` will increment
- Error specific event message `HS_LEN_ERR_EID`

Criticality

None

See also

[HS_ENABLE_CPUHOG_CC](#)

Definition at line 451 of file `hs_msgdefs.h`.

Referenced by `HS_AppPipe()`.

10.8.2.4 HS_DISABLE_EVENTMON_CC

```
#define HS_DISABLE_EVENTMON_CC 5
```

Disable Events Monitor.

Description

Disables the Events Monitor

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentEventMonState](#) will be set to Disabled
- The [HS_DISABLE_EVENTMON_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_ENABLE_EVENTMON_CC](#)

Definition at line 259 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.5 HS_ENABLE_ALIVENESS_CC

```
#define HS_ENABLE_ALIVENESS_CC 6
```

Enable Aliveness Indicator.

Description

Enables the Aliveness Indicator UART output

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentAlivenessState](#) will be set to Enabled
- The [HS_ENABLE_ALIVENESS_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_DISABLE_ALIVENESS_CC](#)

Definition at line 291 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.6 HS_ENABLE_APPMON_CC

```
#define HS_ENABLE_APPMON_CC 2
```

Enable Applications Monitor.

Description

Enables the Applications Monitor

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentAppMonState](#) will be set to Enabled
- The [HS_ENABLE_APPMON_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_DISABLE_APPMON_CC](#)

Definition at line 163 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.7 HS_ENABLE_CPUHOG_CC

```
#define HS_ENABLE_CPUHOG_CC 10
```

Enable CPU Hogging Indicator.

Description

Enables the CPU Hogging Indicator Event Message

Command Structure

`HS_NoArgsCmd_t`

Command Verification

Successful execution of this command may be verified with the following telemetry:

- `HS_HkPacket_t.CmdCount` will increment
- `HS_HkPacket_t.CurrentCPUHogState` will be set to Enabled
- The `HS_ENABLE_CPUHOG_DBG_EID` informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- `HS_HkPacket_t.CmdErrCount` will increment
- Error specific event message `HS_LEN_ERR_EID`

Criticality

None

See also

[HS_DISABLE_CPUHOG_CC](#)

Definition at line 419 of file `hs_msgdefs.h`.

Referenced by `HS_AppPipe()`.

10.8.2.8 HS_ENABLE_EVENTMON_CC

```
#define HS_ENABLE_EVENTMON_CC 4
```

Enable Events Monitor.

Description

Enables the Events Monitor

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.CurrentEventMonState](#) will be set to Enabled
- The [HS_ENABLE_EVENTMON_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_DISABLE_EVENTMON_CC](#)

Definition at line 227 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.9 HS_NOOP_CC

```
#define HS_NOOP_CC 0
```

Noop.

Description

Implements the Noop command that insures the HS task is alive

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- The [HS_NOOP_INF_EID](#) informational event message will be generated when the command is received

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_RESET_CC](#)

Definition at line 100 of file `hs_msgdefs.h`.

Referenced by `HS_AppPipe()`.

10.8.2.10 HS_REPORT_DIAG_CC

```
#define HS_REPORT_DIAG_CC 12
```

Report Util Diagnostics.

Custom command codes must not conflict with those in [hs_msgdefs.h](#)

Description

Reports the Utilization Diagnostics

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- The [HS_UTIL_DIAG_REPORT_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment

Criticality

None

Definition at line 71 of file [hs_custom.h](#).

Referenced by [HS_CustomCommands\(\)](#).

10.8.2.11 HS_RESET_CC

```
#define HS_RESET_CC 1
```

Reset Counters.

Description

Resets the HS housekeeping counters

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will be cleared
- The [HS_RESET_DBG_EID](#) debug event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_NOOP_CC](#)

Definition at line 131 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.12 HS_RESET_RESETS_PERFORMED_CC

```
#define HS_RESET_RESETS_PERFORMED_CC 8
```

Reset Processor Resets Performed Count.

Description

Resets the count of HS performed resets maintained by HS

Command Structure

[HS_NoArgsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.ResetsPerformed](#) will be set to 0
- The [HS_RESET_RESETS_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_SET_MAX_RESETS_CC](#)

Definition at line 355 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.13 HS_SET_MAX_RESETS_CC

```
#define HS_SET_MAX_RESETS_CC 9
```

Set Max Processor Resets Performed Count.

Description

Sets the max allowable count of processor resets to the provided value

Command Structure

[HS_SetMaxResetsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment
- [HS_HkPacket_t.MaxResets](#) will be set to the provided value
- The [HS_SET_MAX_RESETS_DBG_EID](#) informational event message will be generated when the command is executed

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment
- Error specific event message [HS_LEN_ERR_EID](#)

Criticality

None

See also

[HS_RESET_RESETS_PERFORMED_CC](#)

Definition at line 387 of file hs_msgdefs.h.

Referenced by [HS_AppPipe\(\)](#).

10.8.2.14 HS_SET_UTIL_DIAG_CC

```
#define HS_SET_UTIL_DIAG_CC 14
```

Set Utilization Diagnostics Parameter.

Description

Sets the Utilization Diagnostics parameter

Command Structure

[HS_SetUtilDiagCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment

Criticality

None

Definition at line 124 of file hs_custom.h.

Referenced by [HS_CustomCommands\(\)](#).

10.8.2.15 HS_SET_UTIL_PARAMS_CC

```
#define HS_SET_UTIL_PARAMS_CC 13
```

Set Utilization Calibration Parameters.

Description

Sets the Utilization Calibration Parameter

Command Structure

[HS_SetUtilParamsCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- [HS_HkPacket_t.CmdCount](#) will increment

Error Conditions

This command may fail for the following reason(s):

- Command packet length not as expected
- Any parameter is set to 0.

Evidence of failure may be found in the following telemetry:

- [HS_HkPacket_t.CmdErrCount](#) will increment

Criticality

None

Definition at line 98 of file `hs_custom.h`.

Referenced by `HS_CustomCommands()`.

10.9 CFS Health and Safety Version

Macros

- `#define HS_MAJOR_VERSION 2`
Major version number.
- `#define HS_MINOR_VERSION 4`
Minor version number.
- `#define HS_REVISION 99`
Revision number.

10.9.1 Detailed Description

Version Numbers

10.9.2 Macro Definition Documentation

10.9.2.1 HS_MAJOR_VERSION

```
#define HS_MAJOR_VERSION 2
```

Major version number.

Definition at line 34 of file hs_version.h.

Referenced by HS_AppInit(), and HS_NoopCmd().

10.9.2.2 HS_MINOR_VERSION

```
#define HS_MINOR_VERSION 4
```

Minor version number.

Definition at line 35 of file hs_version.h.

Referenced by HS_AppInit(), and HS_NoopCmd().

10.9.2.3 HS_REVISION

```
#define HS_REVISION 99
```

Revision number.

Definition at line 36 of file hs_version.h.

Referenced by HS_AppInit(), and HS_NoopCmd().

10.10 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_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_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)
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)

- `#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)

- `#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.*

10.10.1 Detailed Description

10.10.2 Macro Definition Documentation

10.10.2.1 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 527 of file cfe_error.h.

10.10.2.2 CFE_ES_BAD_ARGUMENT

```
#define CFE_ES_BAD_ARGUMENT ((CFE_Status_t)0xc400000a)
```

Bad Argument.

Bad parameter passed into an ES API.

Definition at line 338 of file cfe_error.h.

10.10.2.3 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 466 of file cfe_error.h.

10.10.2.4 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 544 of file cfe_error.h.

10.10.2.5 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 397 of file cfe_error.h.

10.10.2.6 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 354 of file cfe_error.h.

Referenced by HS_ApplInit().

10.10.2.7 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 448 of file cfe_error.h.

10.10.2.8 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 363 of file cfe_error.h.

10.10.2.9 CFE_ES_CDS_INVALID

```
#define CFE_ES_CDS_INVALID ((CFE_Status_t)0xc4000012)
```

CDS Invalid.

The CDS contents are invalid.

Definition at line 389 of file cfe_error.h.

10.10.2.10 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 372 of file cfe_error.h.

10.10.2.11 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 381 of file cfe_error.h.

10.10.2.12 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 514 of file cfe_error.h.

10.10.2.13 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 503 of file cfe_error.h.

10.10.2.14 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 475 of file cfe_error.h.

10.10.2.15 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 297 of file cfe_error.h.

10.10.2.16 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 421 of file cfe_error.h.

10.10.2.17 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 305 of file cfe_error.h.

10.10.2.18 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 429 of file cfe_error.h.

10.10.2.19 CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK

```
#define CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_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 438 of file `cfe_error.h`.

10.10.2.20 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 346 of file `cfe_error.h`.

10.10.2.21 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 607 of file `cfe_error.h`.

10.10.2.22 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 330 of file `cfe_error.h`.

10.10.2.23 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 322 of file cfe_error.h.

10.10.2.24 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 289 of file cfe_error.h.

10.10.2.25 CFE_ES_ERR_RESOURCEID_NOT_VALID

```
#define CFE_ES_ERR_RESOURCEID_NOT_VALID ((CFE_Status_t)0xc4000001)
```

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 281 of file cfe_error.h.

10.10.2.26 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 314 of file cfe_error.h.

10.10.2.27 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 579 of file cfe_error.h.

10.10.2.28 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 493 of file cfe_error.h.

10.10.2.29 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 405 of file cfe_error.h.

10.10.2.30 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 570 of file cfe_error.h.

10.10.2.31 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 457 of file cfe_error.h.

10.10.2.32 CFE_ES_NO_RESOURCE_IDS_AVAILABLE

```
#define CFE_ES_NO_RESOURCE_IDS_AVAILABLE ((CFE_Status_t)0xc400002B)
```

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 589 of file cfe_error.h.

10.10.2.33 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 618 of file cfe_error.h.

10.10.2.34 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 561 of file cfe_error.h.

10.10.2.35 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 599 of file cfe_error.h.

10.10.2.36 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 484 of file cfe_error.h.

10.10.2.37 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 413 of file cfe_error.h.

10.10.2.38 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 553 of file cfe_error.h.

10.10.2.39 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 536 of file cfe_error.h.

10.10.2.40 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 223 of file cfe_error.h.

10.10.2.41 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 214 of file cfe_error.h.

10.10.2.42 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 205 of file cfe_error.h.

10.10.2.43 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 241 of file `cfe_error.h`.

10.10.2.44 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 249 of file `cfe_error.h`.

10.10.2.45 CFE_EVS_INVALID_PARAMETER

```
#define CFE_EVS_INVALID_PARAMETER ((CFE_Status_t)0xc2000008)
```

Invalid Pointer.

Invalid parameter supplied to EVS command

Definition at line 257 of file `cfe_error.h`.

10.10.2.46 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 268 of file `cfe_error.h`.

10.10.2.47 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 232 of file cfe_error.h.

10.10.2.48 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 197 of file cfe_error.h.

10.10.2.49 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 631 of file cfe_error.h.

10.10.2.50 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 647 of file cfe_error.h.

10.10.2.51 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 639 of file cfe_error.h.

10.10.2.52 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 658 of file cfe_error.h.

10.10.2.53 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 689 of file cfe_error.h.

10.10.2.54 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 747 of file cfe_error.h.

10.10.2.55 CFE_SB_BUFFER_INVALID

```
#define CFE_SB_BUFFER_INVALID ((CFE_Status_t)0xca0000e)
```

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 798 of file cfe_error.h.

10.10.2.56 CFE_SB_INTERNAL_ERR

```
#define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca0000c)
```

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 779 of file cfe_error.h.

10.10.2.57 CFE_SB_MAX_DESTS_MET

```
#define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca0000a)
```

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 769 of file cfe_error.h.

10.10.2.58 CFE_SB_MAX_MSGS_MET

```
#define CFE_SB_MAX_MSGS_MET ((CFE_Status_t)0xca00009)
```

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 757 of file cfe_error.h.

10.10.2.59 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 700 of file `cfe_error.h`.

10.10.2.60 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 734 of file `cfe_error.h`.

10.10.2.61 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 680 of file `cfe_error.h`.

Referenced by `HS_AppMain()`, and `HS_ProcessCommands()`.

10.10.2.62 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 809 of file `cfe_error.h`.

10.10.2.63 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 711 of file [cfe_error.h](#).

10.10.2.64 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 724 of file [cfe_error.h](#).

10.10.2.65 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 671 of file [cfe_error.h](#).

Referenced by [HS_AppMain\(\)](#).

10.10.2.66 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 788 of file [cfe_error.h](#).

10.10.2.67 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 153 of file cfe_error.h.

10.10.2.68 CFE_STATUS_EXTERNAL_RESOURCE_FAIL

```
#define CFE_STATUS_EXTERNAL_RESOURCE_FAIL ((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 165 of file cfe_error.h.

10.10.2.69 CFE_STATUS_NO_COUNTER_INCREMENT

```
#define CFE_STATUS_NO_COUNTER_INCREMENT ((CFE_Status_t)0x48000001)
```

No Counter Increment.

Informational code indicating that a command was processed successfully but that the command counter should *not* be incremented.

Definition at line 126 of file cfe_error.h.

10.10.2.70 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 185 of file cfe_error.h.

10.10.2.71 CFE_STATUS_REQUEST_ALREADY_PENDING

```
#define CFE_STATUS_REQUEST_ALREADY_PENDING ((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 174 of file cfe_error.h.

10.10.2.72 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 144 of file cfe_error.h.

10.10.2.73 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 135 of file cfe_error.h.

10.10.2.74 CFE_SUCCESS

```
#define CFE_SUCCESS ((CFE_Status_t)0)
```

Successful execution.

Operation was performed successfully

Definition at line 118 of file cfe_error.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_AppMain(), HS_AppPipe(), HS_CustomCleanup(), HS_CustomCommands(), HS_CustomInit(), HS_DisableEventMonCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_IdleTask(), HS_MonitorApplications(), HS_MonitorEvent(), HS_ProcessCommands(), HS_ProcessMain(), HS_SbInit(), HS_TblInit(), HS_ValidateAMTable(), HS_ValidateEMTable(), and HS_ValidateMATable().

10.10.2.75 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 1220 of file cfe_error.h.

10.10.2.76 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 1211 of file cfe_error.h.

10.10.2.77 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 1003 of file cfe_error.h.

10.10.2.78 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 1191 of file cfe_error.h.

10.10.2.79 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 1180 of file cfe_error.h.

10.10.2.80 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 1044 of file cfe_error.h.

10.10.2.81 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 955 of file cfe_error.h.

10.10.2.82 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 916 of file cfe_error.h.

10.10.2.83 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 926 of file cfe_error.h.

10.10.2.84 CFE_TBL_ERR_FILE_FOR_WRONG_TABLE

```
#define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc000020)
```

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 1088 of file cfe_error.h.

10.10.2.85 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 1053 of file cfe_error.h.

10.10.2.86 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 983 of file cfe_error.h.

10.10.2.87 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 1079 of file cfe_error.h.

10.10.2.88 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 906 of file cfe_error.h.

10.10.2.89 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 964 of file cfe_error.h.

10.10.2.90 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 823 of file cfe_error.h.

10.10.2.91 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 833 of file [cfe_error.h](#).

10.10.2.92 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 1145 of file [cfe_error.h](#).

10.10.2.93 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 844 of file [cfe_error.h](#).

10.10.2.94 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 973 of file cfe_error.h.

10.10.2.95 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 1097 of file cfe_error.h.

10.10.2.96 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 860 of file cfe_error.h.

10.10.2.97 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 888 of file cfe_error.h.

10.10.2.98 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 946 of file cfe_error.h.

10.10.2.99 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 1061 of file cfe_error.h.

10.10.2.100 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 1070 of file cfe_error.h.

10.10.2.101 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 1119 of file cfe_error.h.

10.10.2.102 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 869 of file cfe_error.h.

10.10.2.103 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 1205 of file cfe_error.h.

10.10.2.104 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 897 of file cfe_error.h.

10.10.2.105 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 1129 of file cfe_error.h.

10.10.2.106 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 1011 of file cfe_error.h.

10.10.2.107 CFE_TBL_INFO_NO_VALIDATION_PENDING

```
#define CFE_TBL_INFO_NO_VALIDATION_PENDING ((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 1035 of file cfe_error.h.

10.10.2.108 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 1169 of file cfe_error.h.

10.10.2.109 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 1019 of file cfe_error.h.

10.10.2.110 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 852 of file cfe_error.h.

10.10.2.111 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 937 of file cfe_error.h.

Referenced by HS_AcquirePointers().

10.10.2.112 CFE_TBL_INFO_VALIDATION_PENDING

```
#define CFE_TBL_INFO_VALIDATION_PENDING ((CFE_Status_t)0x4c000019)
```

Validation Pending

The calling Application should call [CFE_TBL_Validate](#) for the specified table.

Definition at line 1027 of file cfe_error.h.

10.10.2.113 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 1199 of file cfe_error.h.

10.10.2.114 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 1231 of file cfe_error.h.

10.10.2.115 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 879 of file cfe_error.h.

10.10.2.116 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 1156 of file cfe_error.h.

10.10.2.117 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 1107 of file cfe_error.h.

10.10.2.118 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 994 of file cfe_error.h.

10.10.2.119 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 1303 of file cfe_error.h.

10.10.2.120 CFE_TIME_CALLBACK_NOT_REGISTERED

```
#define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000004)
```

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 1294 of file cfe_error.h.

10.10.2.121 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 1258 of file cfe_error.h.

10.10.2.122 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 1246 of file cfe_error.h.

10.10.2.123 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 1273 of file cfe_error.h.

10.10.2.124 CFE_TIME_TOO_MANY_SYNCH_CALLBACKS

```
#define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((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 1284 of file cfe_error.h.

10.11 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.

10.11.1 Detailed Description

10.11.2 Function Documentation

10.11.2.1 `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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.

10.11.2.2 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>CounterId</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.

10.11.2.3 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

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

10.11.2.4 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.

10.12 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.

10.12.1 Detailed Description

10.12.2 Function Documentation

10.12.2.1 [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](#)

10.12.2.2 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](#)

Referenced by [HS_MonitorApplications\(\)](#), and [HS_MonitorEvent\(\)](#).

10.13 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.

10.13.1 Detailed Description

10.13.2 Function Documentation

10.13.2.1 `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	<code>AppID</code>	Identifies the application to be reset.
----	--------------------	---

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.
<code>CFE_SUCCESS</code>	Successful execution.

See also

[CFE_ES_RestartApp](#), [CFE_ES_ReloadApp](#)

Referenced by `HS_MonitorEvent()`.

10.13.2.2 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](#)

10.13.2.3 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	AppID	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_ES_FILE_IO_ERR	File IO Error.
CFE_SUCCESS	Successful execution.

See also

[CFE_ES_ReloadApp](#), [CFE_ES_DeleteApp](#)

Referenced by HS_MonitorApplications(), and HS_MonitorEvent().

10.14 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.

10.14.1 Detailed Description

10.14.2 Function Documentation

10.14.2.1 [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

in	<i>ExitStatus</i>	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 HS_AppMain().

10.14.2.2 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](#)

Referenced by HS_IdleTask().

10.14.2.3 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	RunStatus	Optional pointer to a variable containing the desired run status
---------	-----------	--

Returns

Boolean indicating application should continue running

Return values

true	Application should continue running
false	Application should not continue running

See also

[CFE_ES_ExitApp](#)

Referenced by HS_AppMain().

10.14.2.4 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

in	<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](#)

Referenced by HS_AppMain().

10.14.2.5 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

CFE_SUCCESS	State successfully achieved
CFE_ES_OPERATION_TIMED_OUT	(return value only verified in coverage test) Timeout was reached

See also

[CFE_ES_RunLoop](#)

10.15 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.

10.15.1 Detailed Description

10.15.2 Function Documentation

10.15.2.1 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

out	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null). *AppIdPtr 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](#)

10.15.2.2 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

out	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null).
in	<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](#)

Referenced by `HS_HousekeepingReq()`, `HS_MonitorApplications()`, and `HS_MonitorEvent()`.

10.15.2.3 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

<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_GetAppName](#)

Referenced by [HS_MonitorApplications\(\)](#).

10.15.2.4 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 <i>BufferLength</i> 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 <i>AppName</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_GetAppID](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppInfo](#)

10.15.2.5 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

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_NAME_NOT_FOUND	Resource Name Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_GetLibName](#)

10.15.2.6 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](#)

10.15.2.7 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](#)

10.15.2.8 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

out	<i>ModuleInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>Resource← Id</i>	ID of application or library 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_GetLibInfo](#), [CFE_ES_GetApplInfo](#)

10.15.2.9 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

<code>CFE_PSP_RST_TYPE_POWERON</code>	
<code>CFE_PSP_RST_TYPE_PROCESSOR</code>	

See also

[CFE_ES_GetAppID](#), [CFE_ES_GetAppIDByName](#), [CFE_ES_GetAppName](#), [CFE_ES_GetTaskInfo](#)

10.15.2.10 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

<code>out</code>	<code>TaskIdPtr</code>	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

<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.

10.15.2.11 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

<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_GetTaskID](#), [CFE_ES_GetTaskIDByName](#), [CFE_ES_GetTaskName](#)

Referenced by `HS_HousekeepingReq()`.

10.16 cFE Child Task APIs

Functions

- `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.
- `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.

10.16.1 Detailed Description

10.16.2 Function Documentation

10.16.2.1 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

out	<i>TaskIdPtr</i>	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.
in	<i>TaskName</i>	A pointer to a string containing the desired name of the new task (must not be null). This can be up to OS_MAX_API_NAME characters, including the trailing null.
in	<i>FunctionPtr</i>	A pointer to the function that will be spawned as a new task (must not be null).
in	<i>StackPtr</i>	A pointer to the location where the child task's stack pointer should start. NOTE: Not all underlying operating systems support this parameter. The CFE_ES_TASK_STACK_ALLOCATE constant may be passed to indicate that the stack should be dynamically allocated.
in	<i>StackSize</i>	The number of bytes to allocate for the new task's stack (must not be zero).
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.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_CHILD_TASK_CREATE	Child Task Create Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

See also

[CFE_ES_DeleteChildTask](#), [CFE_ES_ExitChildTask](#)

Referenced by [HS_CustomInit\(\)](#).

10.16.2.2 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>Task</i> <i>Id</i>	The task ID previously obtained when the Child Task was created with the CFE_ES_CreateChildTask API.
----	--------------------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_CHILD_TASK_DELETE	(return value only verified in coverage test) Child Task Delete Error.
CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK	Child Task Delete Passed Main Task.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.

See also

[CFE_ES_CreateChildTask](#), [CFE_ES_ExitChildTask](#)

Referenced by `HS_CustomCleanup()`.

10.16.2.3 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](#)

10.16.2.4 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

out	<i>TaskIdPtr</i>	Pointer to variable that is to receive the Task's ID (must not be null).
in	<i>TaskName</i>	Pointer to null terminated character string containing a Task 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.

See also

[CFE_ES_GetTaskName](#)

Referenced by [HS_HousekeepingReq\(\)](#).

10.16.2.5 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](#)

10.17 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, [uint32](#) 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.

10.17.1 Detailed Description

10.17.2 Function Documentation

10.17.2.1 [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.

10.17.2.2 CFE_ES_CalculateCRC()

```
CFE_Status_t uint32 CFE_ES_CalculateCRC (
    const void * DataPtr,
    size_t DataLength,
    uint32 InputCRC,
    uint32 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: <ul style="list-style-type: none"> • CFE_MISSION_ES_CRC_8 - (Not currently implemented) • CFE_MISSION_ES_CRC_16 - CRC-16/ARC Polynomial: 0x8005 Initialization: 0x0000 Reflect Input/Output: true XorOut: 0x0000 • CFE_MISSION_ES_CRC_32 - (not currently implemented)

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

10.17.2.3 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.

10.17.2.4 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 HS_AppInit(), HS_AppMain(), HS_MonitorApplications(), HS_MonitorEvent(), and HS_MonitorUtilization().

10.18 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)

10.18.1 Detailed Description

10.18.2 Function Documentation

10.18.2.1 `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	<code>Handle</code>	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS .
in	<code>DataToCopy</code>	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

<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_RegisterCDS](#), [CFE_ES_RestoreFromCDS](#)

Referenced by [HS_SetCDSData\(\)](#).

10.18.2.2 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

<i>out</i>	<i>BlockIdPtr</i>	Pointer to variable that is to receive the CDS Block ID (must not be null).
<i>in</i>	<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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_NAME_NOT_FOUND</i>	Resource Name Error.

Return values

CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.

See also

[CFE_ES_GetCDSBlockName](#)

10.18.2.3 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

out	<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.
in	<i>BlockId</i>	Block ID/Handle of CDS registry entry 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>BlockName</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.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.

See also

[CFE_ES_GetCDSBlockIDByName](#)

10.18.2.4 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

out	<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 .
in	<i>BlockSize</i>	The number of bytes needed in the CDS (must not be zero).
in	<i>Name</i>	A pointer to a character string (must not be null) containing an application unique name of CFE_MISSION_ES_CDS_MAX_NAME_LENGTH characters or less.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	The memory block was successfully created in the CDS.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.
CFE_ES_CDS_ALREADY_EXISTS	CDS Already Exists.
CFE_ES_CDS_INVALID_SIZE	CDS Invalid Size.
CFE_ES_CDS_INVALID_NAME	CDS Invalid Name.

Return values

CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_CDS_INVALID	(return value only verified in coverage test) CDS Invalid.

See also

[CFE_ES_CopyToCDS](#), [CFE_ES_RestoreFromCDS](#)

Referenced by HS_AppInit().

10.18.2.5 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

in	<i>Handle</i>	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS .
out	<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

CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Return values

<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<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](#)

Referenced by [HS_ApplInit\(\)](#).

10.19 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.

10.19.1 Detailed Description

10.19.2 Function Documentation

10.19.2.1 `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

<i>out</i>	<i>BufPtr</i>	Pointer to CFE_ES_MemPoolStats_t data structure (must not be null) to be filled with memory statistics. *BufPtr is the Memory Pool Statistics stored in given data structure.
<i>in</i>	<i>Handle</i>	The handle to the memory pool whose statistics are desired.

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_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#)

10.19.2.2 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

<i>out</i>	<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.
<i>in</i>	<i>Handle</i>	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem .
<i>in</i>	<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

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_ERR_MEM_BLOCK_SIZE	Memory Block Size Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_PoolCreate](#), [CFE_ES_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#), [CFE_ES_GetPoolBufInfo](#)

10.19.2.3 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](#)

10.19.2.4 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

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_PoolCreateNoSem](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

10.19.2.5 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

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.
in	<i>NumBlockSizes</i>	The number of different block sizes specified in the <i>BlockSizes</i> array. If set larger than CFE_PLATFORM_ES_POOL_MAX_BUCKETS , CFE_ES_BAD_ARGUMENT will be returned. If BlockSizes is null and NumBlockSizes is 0, NubBlockSizes will be set to CFE_PLATFORM_ES_POOL_MAX_BUCKETS .
in	<i>BlockSizes</i>	Pointer to an array of sizes to be used instead of the default block sizes specified by CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 through CFE_PLATFORM_ES_MAX_BLOCK_SIZE . If the pointer is equal to NULL, the default block sizes are used.
in	<i>UseMutex</i>	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are CFE_ES_USE_MUTEX and CFE_ES_NO_MUTEX

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_NO_RESOURCE_IDS_AVAILABLE</i>	Resource ID is not available.
<i>CFE_STATUS_EXTERNAL_RESOURCE_FAILURE</i>	(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](#)

10.19.2.6 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

<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_PoolCreate](#), [CFE_ES_PoolCreateEx](#), [CFE_ES_GetPoolBuf](#), [CFE_ES_PutPoolBuf](#), [CFE_ES_GetMemPoolStats](#)

10.19.2.7 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](#)

10.19.2.8 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](#)

10.20 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.

10.20.1 Detailed Description

10.20.2 Macro Definition Documentation

10.20.2.1 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 (see section 5.15).

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 1471 of file `cfe_es.h`.

Referenced by HS_AppMain(), and HS_IdleTask().

10.20.2.2 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 (see section 5.15).

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 1490 of file cfe_es.h.

Referenced by HS_AppMain(), and HS_IdleTask().

10.20.3 Function Documentation

10.20.3.1 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 (see section 5.15).

Assumptions, External Events, and Notes:

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](#)

10.21 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.

10.21.1 Detailed Description

10.21.2 Function Documentation

10.21.2.1 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	CounterId	The Counter Id of the newly created counter.
----	-----------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_IncrementGenCounter](#), [CFE_ES_RegisterGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_GetGenCount](#), [CFE_ES_GetGenCounterIDByName](#)

10.21.2.2 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](#)

Referenced by [HS_HousekeepingReq\(\)](#).

10.21.2.3 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

out	<i>CounterIdPtr</i>	Pointer to variable that is to receive the Counter's ID (must not be null).
in	<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

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_NAME_NOT_FOUND	Resource Name Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

[CFE_ES_GetGenCounterName](#)

Referenced by [HS_HousekeepingReq\(\)](#).

10.21.2.4 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

out	<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.
in	<i>CounterId</i>	ID of Counter whose name is being requested.
in	<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

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_GetGenCounterIDByName](#)

10.21.2.5 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_ES_RegisterGenCounter](#), [CFE_ES_DeleteGenCounter](#), [CFE_ES_SetGenCount](#), [CFE_ES_GetGenCount](#),
[CFE_ES_GetGenCounterIDByName](#)

10.21.2.6 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](#)

10.21.2.7 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_Get←GenCount](#), [CFE_ES_GetGenCounterIDByName](#)

10.22 cFE Registration APIs

Functions

- [CFE_Status_t CFE_EVS_Register](#) (const void *Filters, uint16 NumEventFilters, uint16 FilterScheme)
Register an application for receiving event services.

10.22.1 Detailed Description

10.22.2 Function Documentation

10.22.2.1 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

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_FILTER_OVERLOAD	Application Filter Overload.
CFE_EVS_UNKNOWN_FILTER	Unknown Filter.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_ES_BAD_ARGUMENT	Bad Argument.

Referenced by HS_AppInit().

10.23 cFE Send Event APIs

Functions

- `CFE_Status_t CFE_EVS_SendEvent (uint16 EventID, uint16 EventType, const char *Spec,...) OS_PRINTF(3
Generate a software event.`
- `CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithApplID (uint16 EventID, uint16 EventType, CFE_ES_←
ApplId_t ApplID, 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, uint16 EventType, const char *Spec,...) OS_PRINTF(4
Generate a software event with a specific time tag.`

10.23.1 Detailed Description

10.23.2 Function Documentation

10.23.2.1 CFE_EVS_SendEvent()

```
CFE_Status_t CFE_EVS_SendEvent (
    uint16 EventID,
    uint16 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 Event ID is defined and supplied by the application sending the event.
----	----------------	--

Parameters

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.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_EVS_INVALID_PARAMETER	Invalid Pointer.

See also

[CFE_EVS_SendEventWithAppID](#), [CFE_EVS_SendTimedEvent](#)

Referenced by `HS_AcquirePointers()`, `HS_AppInit()`, `HS_AppMain()`, `HS_AppPipe()`, `HS_CustomInit()`, `HS_DisableAlivenessCmd()`, `HS_DisableAppMonCmd()`, `HS_DisableCPUHogCmd()`, `HS_DisableEventMonCmd()`, `HS_EnableAlivenessCmd()`, `HS_EnableAppMonCmd()`, `HS_EnableCPUHogCmd()`, `HS_EnableEventMonCmd()`, `HS_HousekeepingReq()`, `HS_MonitorApplications()`, `HS_MonitorEvent()`, `HS_MonitorUtilization()`, `HS_NoopCmd()`, `HS_ResetCmd()`, `HS_ResetResetsPerformedCmd()`, `HS_SbInit()`, `HS_SetMaxResetsCmd()`, `HS_SetUtilDiagCmd()`, `HS_SetUtilParamsCmd()`, `HS_TblInit()`, `HS_UtilDiagReport()`, `HS_ValidateAMTable()`, `HS_ValidateEMTable()`, `HS_ValidateMATable()`, and `HS_VerifyMsgLength()`.

10.23.2.2 CFE_EVS_SendEventWithAppID()

```
CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (
    uint16 EventID,
```

```
    uint16 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 Event ID 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.

Return values

CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_EVS_INVALID_PARAMETER	Invalid Pointer.

See also

[CFE_EVS_SendEvent](#), [CFE_EVS_SendTimedEvent](#)

10.23.2.3 CFE_EVS_SendTimedEvent()

```
CFE_Status_t CFE_Status_t CFE_Status_t CFE_EVS_SendTimedEvent (
    CFE_TIME_SysTime_t Time,
    uint16 EventID,
    uint16 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>Event ID</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

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_EVS_APP_NOT_REGISTERED</i>	Application Not Registered.
<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](#)

10.24 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.

10.24.1 Detailed Description

10.24.2 Function Documentation

10.24.2.1 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](#)

10.24.2.2 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 <i>Event ID</i> 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](#)

10.25 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.

10.25.1 Detailed Description

10.25.2 Function Documentation

10.25.2.1 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	<i>Hdr</i>	Pointer to a variable of type CFE_FS_Header_t that will be cleared and initialized
in	<i>Description</i>	Initializes Header's Description (must not be null)
in	<i>SubType</i>	Initializes Header's SubType

See also

[CFE_FS_WriteHeader](#)

10.25.2.2 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

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.
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.

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](#)

10.25.2.3 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_FAILURE	(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](#).

10.25.2.4 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

<i>CFE_FS_BAD_ARGUMENT</i>	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](#)

10.26 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_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t *Meta)`
Query if a background file write request is currently pending.

10.26.1 Detailed Description

10.26.2 Function Documentation

10.26.2.1 CFE_FS_BackgroundFileDumpIsPending()

```
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

<code>in, out</code>	<code>Meta</code>	The background file write persistent state object (must not be null)
----------------------	-------------------	--

Returns

boolean value indicating if request is already pending

Return values

<i>true</i>	if request is pending
<i>false</i>	if request is not pending

10.26.2.2 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

<i>in, out</i>	<i>Meta</i>	The background file write persistent state object (must not be null)
----------------	-------------	--

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

<i>CFE_FS_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_FS_INVALID_PATH</i>	Invalid Path.
<i>CFE_STATUS_REQUEST_ALREADY_PENDING</i>	Request already pending.
<i>CFE_SUCCESS</i>	Successful execution.

10.26.2.3 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

in	<i>OriginalPath</i>	The original path (must not be null)
out	<i>FileNameOnly</i>	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.

10.26.2.4 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
-------------------	--

10.26.2.5 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
-------------------	--

10.26.2.6 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**

<i>out</i>	<i>OutputBuffer</i>	Buffer to store result (must not be null).
<i>in</i>	<i>InputName</i>	A null terminated input string (must not be null).
<i>in</i>	<i>OutputBufSize</i>	Maximum Size of output buffer (must not be zero).
<i>in</i>	<i>FileCategory</i>	The generalized category of file (implies default path/extension)

ReturnsExecution status, see [cFE Return Code Defines](#)**10.26.2.7 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 InputBfffer 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.

10.27 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.

10.27.1 Detailed Description

10.27.2 Function Documentation

10.27.2.1 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 HS_SblInit().

10.28 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)

10.28.1 Detailed Description

10.28.2 Function Documentation

10.28.2.1 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.28.2.2 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.28.2.3 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

<i>in</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>out</i>	<i>Version</i>	Header version (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.

10.28.2.4 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

10.28.2.5 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

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.6 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.

10.28.2.7 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 HS_ValidateMATable(), and HS_VerifyMsgLength().

10.28.2.8 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

<i>in</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>out</i>	<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.

10.28.2.9 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.

10.28.2.10 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.

10.28.2.11 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.

10.28.2.12 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

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.13 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

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>SeqCnt</i>	Sequence count

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.14 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

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>Size</i>	Total message size

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.

10.28.2.15 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

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<i>Type</i>	Message type

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 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.

10.29.1 Detailed Description

10.29.2 Function Documentation

10.29.2.1 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	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Version</i>	EDS Version (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.

10.29.2.2 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

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Endian</i>	Endian (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.

10.29.2.3 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

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>PlayFlag</i>	Playback 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.

10.29.2.4 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.5 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

<i>in</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>out</i>	<i>System</i>	System (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_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

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>Version</i>	EDS 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.

10.29.2.7 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

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>Endian</i>	Endian

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.

10.29.2.8 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_MSG_BAD_ARGUMENT</i>	Error - bad argument.

10.29.2.9 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.

10.29.2.10 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

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>System</i>	System

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 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.

10.30.1 Detailed Description

10.30.2 Function Documentation

10.30.2.1 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

in, out	MsgPtr	A pointer to the buffer that contains 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.

10.30.2.2 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

<i>in</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>out</i>	<i>FcnCode</i>	The function code 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.

Referenced by HS_AppPipe(), HS_CustomCommands(), and HS_VerifyMsgLength().

10.30.2.3 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

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.
CFE_MSG_WRONG_MSG_TYPE	Error - wrong type.

10.30.2.4 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

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<i>in</i>	<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.

10.30.2.5 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

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the message (must not be null).
<i>in</i>	<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

<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.

10.30.2.6 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.
out	<i>IsValid</i>	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

<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 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.

10.31.1 Detailed Description

10.31.2 Function Documentation

10.31.2.1 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	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
out	<code>MsgId</code>	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 `HS_AppPipe()`, `HS_ValidateMATable()`, and `HS_VerifyMsgLength()`.

10.31.2.2 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

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.31.2.3 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.

10.32 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 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.

10.32.1 Detailed Description

10.32.2 Function Documentation

10.32.2.1 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

<i>out</i>	<i>PipeIdPtr</i>	A pointer to a variable of type CFE_SB_PipeId_t (must not be null), which will be filled in with the pipe ID information by the CFE_SB_CreatePipe routine. *PipeIdPtr is the identifier for the created pipe.
<i>in</i>	<i>Depth</i>	The maximum number of messages that will be allowed on this pipe at one time.
<i>in</i>	<i>PipeName</i>	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 OS_MAX_API_NAME (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 [HS_SbInit\(\)](#).

10.32.2.2 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_GetPipeldByName](#)

10.32.2.3 CFE_SB_GetPipeldByName()

```
CFE_Status_t CFE_SB_GetPipeldByName (
    CFE_SB_Pipeld_t * PipeldPtr,
    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>PipeldPtr</i>	The Pipeld 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](#)

10.32.2.4 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](#)

10.32.2.5 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_PIP←EOPTS_IGNOREMINE](#)

10.32.2.6 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.

10.32.2.7 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_PIPE_OPTS_IGNOREMINE](#)

10.33 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.

10.33.1 Detailed Description

10.33.2 Function Documentation

10.33.2.1 `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	<code>MsgId</code>	The message ID of the message to be subscribed to.
in	<code>PipeId</code>	The pipe ID of the pipe the subscribed message should be sent to.

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_SubscribeEx](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_Unsubscribe](#), [CFE_SB_UnsubscribeLocal](#)

Referenced by `HS_SblInit()`.

10.33.2.2 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](#)

Referenced by `HS_AppMain()`, and `HS_EnableEventMonCmd()`.

10.33.2.3 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

<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_SubscribeEx](#), [CFE_SB_Unsubscribe](#), [CFE_SB_UnsubscribeLocal](#)

10.33.2.4 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](#)

Referenced by `HS_AcquirePointers()`, and `HS_DisableEventMonCmd()`.

10.33.2.5 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

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

[CFE_SB_Subscribe](#), [CFE_SB_SubscribeEx](#), [CFE_SB_SubscribeLocal](#), [CFE_SB_Unsubscribe](#)

10.34 cFE Send/Receive Message APIs

Functions

- [CFE_Status_t CFE_SB_TransmitMsg](#) (const [CFE_MSG_Message_t](#) *MsgPtr, bool IncrementSequenceCount)
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.

10.34.1 Detailed Description

10.34.2 Function Documentation

10.34.2.1 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	<i>BufPtr</i>	A pointer to the software bus buffer to receive to (must not be null). 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 receipt of a message, *BufPtr 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 *BufPtr is valid only until the next call to CFE_SB_ReceiveBuffer for the same pipe.
in	<i>PipeId</i>	The pipe ID of the pipe containing the message to be obtained.
in	<i>TimeOut</i>	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to CFE_SB_POLL for a non-blocking receive or CFE_SB_PEND_FOREVER to wait forever for a message to arrive.

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_TIME_OUT</i>	Time Out.
<i>CFE_SB_PIPE_RD_ERR</i>	(return value only verified in coverage test) Pipe Read Error.
<i>CFE_SB_NO_MESSAGE</i>	No Message.

Referenced by HS_AppMain(), and HS_ProcessCommands().

10.34.2.2 CFE_SB_TransmitMsg()

```
CFE_Status_t CFE_SB_TransmitMsg (
    const CFE_MSG_Message_t * MsgPtr,
    bool IncrementSequenceCount )
```

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.

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.

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>IncrementSequenceCount</i>	Boolean to increment the internally tracked sequence count and update the message if the buffer contains a telemetry 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 HS_HousekeepingReq(), HS_MonitorApplications(), and HS_MonitorEvent().

10.35 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 \(CFE_SB_Buffer_t *BufPtr, bool IncrementSequenceCount\)](#)
Transmit a buffer.

10.35.1 Detailed Description

10.35.2 Function Documentation

10.35.2.1 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	MsgSize	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\(\)](#).

10.35.2.2 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.

10.35.2.3 CFE_SB_TransmitBuffer()

```
CFE_Status_t CFE_SB_TransmitBuffer (
    CFE_SB_Buffer_t * BufPtr,
    bool IncrementSequenceCount )
```

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.

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>IncrementSequenceCount</i>	Boolean to increment the internally tracked sequence count and update the message if the buffer contains a telemetry 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.

10.36 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.

10.36.1 Detailed Description

10.36.2 Function Documentation

10.36.2.1 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	<i>MsgPtr</i>	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.

10.36.2.2 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	MsgPtr	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 MsgPtr argument is not valid.
---	--

10.36.2.3 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)
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.
-------------------------------------	---------------

10.36.2.4 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.
-------------------------------------	---------------

10.36.2.5 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).

10.36.2.6 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 [HS_HousekeepingReq\(\)](#).

10.37 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_t](#) [CFE_SB_ValueToMsgId](#) ([CFE_SB_MsgId_Atom_t](#) MsgIdValue)
Converts a normal integer into a [CFE_SB_MsgId_t](#).

10.37.1 Detailed Description

10.37.2 Function Documentation

10.37.2.1 [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

Referenced by HS_ValidateMATable().

10.37.2.2 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

<code>true</code>	Message IDs are Equal
<code>false</code>	Message IDs are not Equal

Definition at line 766 of file `cfe_sb.h`.

References `CFE_SB_MSGID_UNWRAP_VALUE`.

10.37.2.3 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 797 of file cfe_sb.h.

References CFE_SB_MSGID_UNWRAP_VALUE.

Referenced by HS_AppPipe(), HS_ValidateMATable(), and HS_VerifyMsgLength().

10.37.2.4 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 826 of file cfe_sb.h.

References CFE_SB_MSGID_C.

Referenced by HS_AcquirePointers(), HS_AppMain(), HS_DisableEventMonCmd(), HS_EnableEventMonCmd(), and HS_SbInit().

10.38 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.

10.38.1 Detailed Description

10.38.2 Macro Definition Documentation

10.38.2.1 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.

10.39 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.

10.39.1 Detailed Description

10.39.2 Function Documentation

10.39.2.1 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.

Parameters

in	<i>Size</i>	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 updated.
Generated by Doxygen		

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 `HS_TblInit()`.

10.39.2.2 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

out	<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.
in	<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](#)

10.39.2.3 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](#)

10.40 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.

10.40.1 Detailed Description

10.40.2 Function Documentation

10.40.2.1 `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

<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.
<i>CFE_TBL_INFO_DUMP_PENDING</i>	Dump Pending.

See also

[CFE_TBL_Manage](#)

10.40.2.2 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"> • <i>CFE_TBL_SRC_FILE</i> - 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. • <i>CFE_TBL_SRC_ADDRESS</i> - 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 a assumed to be open the same size specified in the CFE_TBL_Register function Size parameter.

Parameters

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.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.
CFE_TBL_ERR_DUMP_ONLY	Dump Only Error.
CFE_TBL_ERR_ILLEGAL_SRC_TYPE	Illegal Source Type.
CFE_TBL_ERR_LOAD_IN_PROGRESS	Load In Progress.
CFE_TBL_ERR_LOAD_INCOMPLETE	Load Incomplete.
CFE_TBL_ERR_NO_BUFFER_AVAIL	No Buffer Available.
CFE_TBL_ERR_ACCESS	
CFE_TBL_ERR_FILE_TOO_LARGE	File Too Large.
CFE_TBL_ERR_BAD_CONTENT_ID	Bad Content ID.
CFE_TBL_ERR_BAD_SUBTYPE_ID	Bad Subtype ID.
CFE_TBL_ERR_NO_STD_HEADER	No Standard Header.
CFE_TBL_ERR_NO_TBL_HEADER	No Table Header.
CFE_TBL_ERR_PARTIAL_LOAD	Partial Load Error.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

[CFE_TBL_Update](#), [CFE_TBL_Validate](#), [CFE_TBL_Manage](#)

Referenced by [HS_TblInit\(\)](#).

10.40.2.3 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

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_INFO_DUMP_PENDING	Dump Pending.
CFE_TBL_INFO_UPDATE_PENDING	Update Pending.
CFE_TBL_INFO_VALIDATION_PENDING	

See also

[CFE_TBL_Update](#), [CFE_TBL_Validate](#), [CFE_TBL_Load](#), [CFE_TBL_DumpToBuffer](#)

Referenced by HS_AcquirePointers().

10.40.2.4 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

in	<i>TblHandle</i>	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](#)

10.40.2.5 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_NO_UPDATE_PENDING</i>	No Update Pending.
<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_Load](#), [CFE_TBL_Validate](#), [CFE_TBL_Manage](#)

10.40.2.6 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_NO_VALIDATION_PENDING</i>	

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.

See also

[CFE_TBL_Update](#), [CFE_TBL_Manage](#), [CFE_TBL_Load](#)

10.41 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.

10.41.1 Detailed Description

10.41.2 Function Documentation

10.41.2.1 `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

out	<i>TblPtr</i>	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. *TblPtr is the address of the first byte of data associated with the specified table.
in	<i>TblHandle</i>	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share , 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 HS_AcquirePointers().

10.41.2.2 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](#)

10.41.2.3 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

in	<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

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_NEVER_LOADED	Never Loaded.

See also

[CFE_TBL_GetAddress](#), [CFE_TBL_GetAddresses](#), [CFE_TBL_ReleaseAddresses](#)

Referenced by `HS_AcquirePointers()`.

10.41.2.4 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

in	<i>NumTables</i>	Size of TblHandles array.
in	<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

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_NEVER_LOADED	Never Loaded.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

[CFE_TBL_GetAddress](#), [CFE_TBL_ReleaseAddress](#), [CFE_TBL_GetAddresses](#)

10.42 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.

10.42.1 Detailed Description

10.42.2 Function Documentation

10.42.2.1 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_ERR_INVALID_NAME</i>	Invalid Name.
<i>CFE_TBL_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE_TBL_GetStatus](#)

10.42.2.2 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

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATE_PENDING</i>	Update Pending.

Return values

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](#)

10.42.2.3 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 <i>MsgId</i> and <i>Command Code</i> to be used for all table management notifications.
Generated by Doxygen		

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_Register](#)

10.43 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.

10.43.1 Detailed Description

10.43.2 Macro Definition Documentation

10.43.2.1 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`.

10.43.2.2 CFE_TBL_OPT_CRITICAL

```
#define CFE_TBL_OPT_CRITICAL (0x0008)
```

Critical table.

Definition at line 63 of file cfe_tbl_api_typedefs.h.

10.43.2.3 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.

10.43.2.4 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.

10.43.2.5 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.

Referenced by HS_TblInit().

10.43.2.6 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.

10.43.2.7 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`.

10.43.2.8 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`.

10.43.2.9 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`.

10.43.2.10 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`.

10.43.2.11 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`.

10.43.2.12 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`.

10.43.2.13 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`.

10.44 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.

10.44.1 Detailed Description

10.44.2 Function Documentation

10.44.2.1 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](#)

10.44.2.2 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](#)

10.44.2.3 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_GetMETsubsecs](#), [CFE_TIME_MET2SCTime](#)

10.44.2.4 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](#)

10.44.2.5 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](#)

10.44.2.6 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](#)

10.45 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.

10.45.1 Detailed Description

10.45.2 Function Documentation

10.45.2.1 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](#)

10.45.2.2 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](#)

10.45.2.3 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 SCTF, 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](#)

10.45.2.4 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 SCTF, 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](#)

10.46 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.

10.46.1 Detailed Description

10.46.2 Function Documentation

10.46.2.1 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	<code>Time1</code>	The first time to be added.
in	<code>Time2</code>	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](#)

10.46.2.2 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	TimeA	The first time to compare.
in	TimeB	The second time to compare.

Returns

The result of comparing the two times.

Return values

CFE_TIME_EQUAL	The two specified times are considered to be equal.
CFE_TIME_A_GT↔_B	The first specified time is considered to be after the second specified time.
CFE_TIME_A_LT↔_B	The first specified time is considered to be before the second specified time.

See also

[CFE_TIME_Add](#), [CFE_TIME_Subtract](#)

10.46.2.3 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	Time1	The base time.
in	Time2	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](#)

10.47 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.

10.47.1 Detailed Description

10.47.2 Function Documentation

10.47.2.1 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	METTime	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](#)

10.47.2.2 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	MicroSeconds	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](#),

10.47.2.3 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](#),

10.48 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.

10.48.1 Detailed Description

10.48.2 Function Documentation

10.48.2.1 [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 space-craft 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](#)

10.48.2.2 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](#)

10.48.2.3 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 space-craft 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	<i>NewTime</i>	The MET value at the next (or previous) 1 Hz tone signal.
----	----------------	---

See also

[CFE_TIME_ExternalTone](#), [CFE_TIME_ExternalMET](#), [CFE_TIME_ExternalGPS](#)

10.48.2.4 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](#)

10.48.2.5 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](#)

Referenced by HS_CustomInit().

10.48.2.6 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_RegisterSynchCallback](#)

Referenced by HS_CustomCleanup().

10.49 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.

10.49.1 Detailed Description

10.49.2 Function Documentation

10.49.2.1 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.

10.49.2.2 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.

10.50 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 }
- 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) }

10.50.1 Detailed Description

10.50.2 Enumeration Type Documentation

10.50.2.1 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	

Definition at line 48 of file `cfe_core_resourceid_basevalues.h`.

10.50.2.2 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	

Definition at line 81 of file cfe_core_resourceid_basevalues.h.

10.51 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.

10.51.1 Detailed Description

10.51.2 Macro Definition Documentation

10.51.2.1 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 715 of file cfe_time_msg.h.

10.51.2.2 CFE_TIME_FLAG_ADDADJ

```
#define CFE_TIME_FLAG_ADDADJ 0x0200
```

One time STCF Adjustment is to be done in positive direction.

Definition at line 714 of file cfe_time_msg.h.

10.51.2.3 CFE_TIME_FLAG_ADDTCL

```
#define CFE_TIME_FLAG_ADDTCL 0x0080
```

Time Client Latency is applied in a positive direction.

Definition at line 716 of file cfe_time_msg.h.

10.51.2.4 CFE_TIME_FLAG_CLKSET

```
#define CFE_TIME_FLAG_CLKSET 0x8000
```

The spacecraft time has been set.

Definition at line 708 of file cfe_time_msg.h.

10.51.2.5 CFE_TIME_FLAG_CMDFLY

```
#define CFE_TIME_FLAG_CMDFLY 0x0400
```

This instance of Time Services was commanded into flywheel mode.

Definition at line 713 of file cfe_time_msg.h.

10.51.2.6 CFE_TIME_FLAG_FLYING

```
#define CFE_TIME_FLAG_FLYING 0x4000
```

This instance of Time Services is flywheeling.

Definition at line 709 of file cfe_time_msg.h.

10.51.2.7 CFE_TIME_FLAG_GDTONE

```
#define CFE_TIME_FLAG_GDTONE 0x0020
```

The tone received is good compared to the last tone received.

Definition at line 718 of file cfe_time_msg.h.

10.51.2.8 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 719 of file cfe_time_msg.h.

10.51.2.9 CFE_TIME_FLAG_SERVER

```
#define CFE_TIME_FLAG_SERVER 0x0040
```

This instance of Time Services is a Time Server.

Definition at line 717 of file cfe_time_msg.h.

10.51.2.10 CFE_TIME_FLAG_SIGPRI

```
#define CFE_TIME_FLAG_SIGPRI 0x1000
```

The clock signal is set to "primary".

Definition at line 711 of file cfe_time_msg.h.

10.51.2.11 CFE_TIME_FLAG_SRCINT

```
#define CFE_TIME_FLAG_SRCINT 0x2000
```

The clock source is set to "internal".

Definition at line 710 of file cfe_time_msg.h.

10.51.2.12 CFE_TIME_FLAG_SRVFLY

```
#define CFE_TIME_FLAG_SRVFLY 0x0800
```

The Time Server is in flywheel mode.

Definition at line 712 of file cfe_time_msg.h.

10.51.2.13 CFE_TIME_FLAG_UNUSED

```
#define CFE_TIME_FLAG_UNUSED 0x000F
```

Reserved flags - should be zero.

Definition at line 721 of file cfe_time_msg.h.

10.52 OSAL Semaphore State Defines

Macros

- `#define OS_SEM_FULL 1`
Semaphore full state.
- `#define OS_SEM_EMPTY 0`
Semaphore empty state.

10.52.1 Detailed Description

10.52.2 Macro Definition Documentation

10.52.2.1 OS_SEM_EMPTY

```
#define OS_SEM_EMPTY 0
```

Semaphore empty state.

Definition at line 35 of file osapi-binsem.h.

10.52.2.2 OS_SEM_FULL

```
#define OS_SEM_FULL 1
```

Semaphore full state.

Definition at line 34 of file osapi-binsem.h.

10.53 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.

10.53.1 Detailed Description

10.53.2 Function Documentation

10.53.2.1 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if sen 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 binary semaphore
<i>OS_SEM_FAILURE</i>	if the OS call failed (return value only verified in coverage test)

10.53.2.2 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)

10.53.2.3 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

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a binary semaphore
OS_SEM_FAILURE	if an unspecified failure occurs (return value only verified in coverage test)

10.53.2.4 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	<i>sem_id</i>	will be set to the ID of the existing resource
in	<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

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	is semid or sem_name are NULL pointers
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NAME_NOT_FOUND	if the name was not found in the table

10.53.2.5 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	<i>sem_id</i>	The object ID to operate on
out	<i>bin_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 bin_prop pointer is null

10.53.2.6 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)

10.53.2.7 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.
<i>OS_ERR_INVALID_ID</i>	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)

10.53.2.8 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

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_SEM_TIMEOUT</code>	if semaphore was not relinquished in time
<code>OS_ERR_INVALID_ID</code>	if the ID passed in is not a valid semaphore ID
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

10.54 OSAL BSP low level access APIs

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)`

10.54.1 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

10.54.2 Function Documentation

10.54.2.1 OS_BSP_GetArgC()

```
uint32 OS_BSP_GetArgC (
    void )
```

10.54.2.2 OS_BSP_GetArgV()

```
char* const* OS_BSP_GetArgV (
    void )
```

10.54.2.3 OS_BSP_GetResourceTypeConfig()

```
uint32 OS_BSP_GetResourceTypeConfig (
    uint32 ResourceType )
```

10.54.2.4 OS_BSP_SetExitCode()

```
void OS_BSP_SetExitCode (
    int32 code )
```

10.54.2.5 OS_BSP_SetResourceTypeConfig()

```
void OS_BSP_SetResourceTypeConfig (
    uint32 ResourceType,
    uint32 ConfigOptionValue )
```

10.55 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.
- static `int64 OS_TimeGetTotalSeconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to whole number of seconds.
- static `int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to millisecond units.
- static `int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to microsecond units.
- static `int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to nanosecond units.
- 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)`
Get microseconds portion (fractional part only) from an `OS_time_t` object.
- 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.

10.55.1 Detailed Description

10.55.2 Function Documentation

10.55.2.1 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	<i>time_struct</i>	An OS_time_t that will be set to the current time (must not be null)
-----	--------------------	--

Returns

Get local time status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if time_struct is null

10.55.2.2 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	<i>time_struct</i>	An OS_time_t 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

10.55.2.3 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	<code>time1</code>	The first interval
in	<code>time2</code>	The second interval

Returns

The sum of the two intervals (time1 + time2)

Definition at line 388 of file osapi-clock.h.

References OS_time_t::ticks.

10.55.2.4 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 323 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, OS_TIME_TICKS_PER_USEC, and OS_time_t::ticks.

10.55.2.5 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 347 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC, OS_TIME_TICKS_PER_SECOND, and OS_time_t::ticks.

10.55.2.6 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	seconds	Whole number of seconds
in	nanoseconds	Number of nanoseconds (fractional part only)

Returns

The input arguments represented as an [OS_time_t](#) interval

Definition at line 299 of file osapi-clock.h.

References [OS_TIME_TICK_RESOLUTION_NS](#), [OS_TIME_TICKS_PER_SECOND](#), and [OS_time_t::ticks](#).

10.55.2.7 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 370 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and OS_time_t::ticks.

10.55.2.8 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 191 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().

10.55.2.9 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_usecs" field of POSIX-style "struct timeval" values.

See also

[OS_TimeGetTotalSeconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

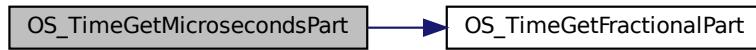
Returns

Number of microseconds in time interval

Definition at line 259 of file osapi-clock.h.

References OS_TIME_TICKS_PER_USEC, and [OS_TimeGetFractionalPart\(\)](#).

Here is the call graph for this function:



10.55.2.10 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	<i>tm</i>	Time interval value
----	-----------	---------------------

Returns

Number of milliseconds in time interval

Definition at line 234 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.55.2.11 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	tm	Time interval value
----	----	---------------------

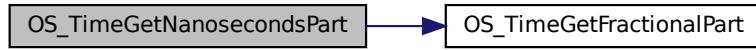
Returns

Number of nanoseconds in time interval

Definition at line 278 of file osapi-clock.h.

References OS_TIME_TICK_RESOLUTION_NS, and OS_TimeGetFractionalPart().

Here is the call graph for this function:



10.55.2.12 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	tm	Time interval value
----	----	---------------------

Returns

Fractional/subsecond portion of time interval as 32-bit fixed point value

Definition at line 210 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and [OS_TimeGetFractionalPart\(\)](#).

Here is the call graph for this function:



10.55.2.13 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.

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of microseconds in time interval

Definition at line 158 of file osapi-clock.h.

References OS_TIME_TICKS_PER_USEC, and OS_time_t::ticks.

10.55.2.14 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.

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of milliseconds in time interval

Definition at line 144 of file osapi-clock.h.

References OS_TIME_TICKS_PER_MSEC, and OS_time_t::ticks.

10.55.2.15 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.

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of microseconds in time interval

Definition at line 176 of file osapi-clock.h.

References OS_TIME_TICK_RESOLUTION_NS, and OS_time_t::ticks.

10.55.2.16 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\(\)](#)

Parameters

in	tm	Time interval value
----	----	---------------------

Returns

Whole number of seconds in time interval

Definition at line 130 of file osapi-clock.h.

References OS_TIME_TICKS_PER_SECOND, and OS_time_t::ticks.

10.55.2.17 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 402 of file osapi-clock.h.

References OS_time_t::ticks.

10.56 OSAL Core Operation APIs

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.

10.56.1 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

10.56.2 Function Documentation

10.56.2.1 [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

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERROR</code>	Failed execution. (return value only verified in coverage test)

10.56.2.2 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.

10.56.2.3 OS_Application_Run()

```
void OS_Application_Run (
    void )
```

Application run.

Run abstraction such that the same BSP can be used for operations and testing.

10.56.2.4 OS_Application_Startup()

```
void OS_Application_Startup (
    void )
```

Application startup.

Startup abstraction such that the same BSP can be used for operations and testing.

10.56.2.5 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.

10.56.2.6 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	<i>flag</i>	set to true to initiate shutdown, false to cancel
----	-------------	---

10.56.2.7 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.

10.56.2.8 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.

10.56.2.9 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	handler	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

10.57 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.

10.57.1 Detailed Description

10.57.2 Function Documentation

10.57.2.1 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	<i>sem_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<i>sem_name</i>	the name of the new resource to create (must not be null)
in	<i>sem_initial_value</i>	the initial value of the counting semaphore
in	<i>options</i>	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 OS_MAX_API_NAME
<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)

10.57.2.2 OS_CountSemDelete()

```
int32 OS_CountSemDelete (
    osal_id_t sem_id )
```

Deletes the specified counting Semaphore.

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 counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

10.57.2.3 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

out	<i>sem_id</i>	will be set to the ID of the existing resource
in	<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>	is 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

10.57.2.4 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

in	<i>sem_id</i>	The object ID to operate on
out	<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

10.57.2.5 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)

10.57.2.6 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)

10.57.2.7 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)

10.58 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.

10.58.1 Detailed Description

10.58.2 Function Documentation

10.58.2.1 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

10.58.2.2 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

10.58.2.3 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

10.58.2.4 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

10.58.2.5 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

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if path is NULL
OS_FS_ERR_PATH_TOO_LONG	if the path is too long to be stored locally
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_ERROR	if the OS call fails (return value only verified in coverage test)

10.58.2.6 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

Return values

<i>OS_FS_ERR_PATH_TOO_LONG</i>	
<i>OS_ERROR</i>	if the directory remove operation failed (return value only verified in coverage test)

10.59 OSAL Return Code Defines

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)

- `#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.

10.59.1 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...
}
```

10.59.2 Macro Definition Documentation

10.59.2.1 OS_ERR_BAD_ADDRESS

```
#define OS_ERR_BAD_ADDRESS (-34)
```

Bad address.

Definition at line 110 of file osapi-error.h.

10.59.2.2 OS_ERR_FILE

```
#define OS_ERR_FILE (-27)
```

File error.

Definition at line 103 of file osapi-error.h.

10.59.2.3 OS_ERR_INCORRECT_OBJ_STATE

```
#define OS_ERR_INCORRECT_OBJ_STATE (-35)
```

Incorrect object state.

Definition at line 111 of file osapi-error.h.

10.59.2.4 OS_ERR_INCORRECT_OBJ_TYPE

```
#define OS_ERR_INCORRECT_OBJ_TYPE (-36)
```

Incorrect object type.

Definition at line 112 of file osapi-error.h.

10.59.2.5 OS_ERR_INVALID_ARGUMENT

```
#define OS_ERR_INVALID_ARGUMENT (-42)
```

Invalid argument value (other than ID or size)

Definition at line 117 of file osapi-error.h.

10.59.2.6 OS_ERR_INVALID_ID

```
#define OS_ERR_INVALID_ID (-16)
```

Invalid ID.

Definition at line 98 of file osapi-error.h.

10.59.2.7 OS_ERR_INVALID_PRIORITY

```
#define OS_ERR_INVALID_PRIORITY (-19)
```

Invalid priority.

Definition at line 101 of file osapi-error.h.

10.59.2.8 OS_ERR_INVALID_SIZE

```
#define OS_ERR_INVALID_SIZE (-40)
```

Invalid Size.

Definition at line 115 of file osapi-error.h.

10.59.2.9 OS_ERR_NAME_NOT_FOUND

```
#define OS_ERR_NAME_NOT_FOUND (-17)
```

Name not found.

Definition at line 99 of file osapi-error.h.

10.59.2.10 OS_ERR_NAME_TAKEN

```
#define OS_ERR_NAME_TAKEN (-15)
```

Name taken.

Definition at line 97 of file osapi-error.h.

10.59.2.11 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 95 of file osapi-error.h.

10.59.2.12 OS_ERR_NO_FREE_IDS

```
#define OS_ERR_NO_FREE_IDS (-14)
```

No free IDs.

Definition at line 96 of file osapi-error.h.

10.59.2.13 OS_ERR_NOT_IMPLEMENTED

```
#define OS_ERR_NOT_IMPLEMENTED (-28)
```

Not implemented.

Definition at line 104 of file osapi-error.h.

10.59.2.14 OS_ERR_OBJECT_IN_USE

```
#define OS_ERR_OBJECT_IN_USE (-33)
```

Object in use.

Definition at line 109 of file osapi-error.h.

10.59.2.15 OS_ERR_OPERATION_NOT_SUPPORTED

```
#define OS_ERR_OPERATION_NOT_SUPPORTED (-38)
```

Requested operation not support on supplied object(s)

Definition at line 114 of file osapi-error.h.

10.59.2.16 OS_ERR_OUTPUT_TOO_LARGE

```
#define OS_ERR_OUTPUT_TOO_LARGE (-41)
```

Size of output exceeds limit.

Definition at line 116 of file osapi-error.h.

10.59.2.17 OS_ERR_SEM_NOT_FULL

```
#define OS_ERR_SEM_NOT_FULL (-18)
```

Semaphore not full.

Definition at line 100 of file osapi-error.h.

10.59.2.18 OS_ERR_STREAM_DISCONNECTED

```
#define OS_ERR_STREAM_DISCONNECTED (-37)
```

Stream disconnected.

Definition at line 113 of file osapi-error.h.

10.59.2.19 OS_ERROR

```
#define OS_ERROR (-1)
```

Failed execution.

Definition at line 83 of file osapi-error.h.

10.59.2.20 OS_ERROR_ADDRESS_MISALIGNED

```
#define OS_ERROR_ADDRESS_MISALIGNED (-3)
```

Address misalignment.

Definition at line 85 of file osapi-error.h.

10.59.2.21 OS_ERROR_TIMEOUT

```
#define OS_ERROR_TIMEOUT (-4)
```

Error timeout.

Definition at line 86 of file osapi-error.h.

10.59.2.22 OS_FS_ERR_DEVICE_NOT_FREE

```
#define OS_FS_ERR_DEVICE_NOT_FREE (-107)
```

FS device not free.

Definition at line 130 of file osapi-error.h.

10.59.2.23 OS_FS_ERR_DRIVE_NOT_CREATED

```
#define OS_FS_ERR_DRIVE_NOT_CREATED (-106)
```

FS drive not created.

Definition at line 129 of file osapi-error.h.

10.59.2.24 OS_FS_ERR_NAME_TOO_LONG

```
#define OS_FS_ERR_NAME_TOO_LONG (-104)
```

FS name too long.

Definition at line 128 of file osapi-error.h.

10.59.2.25 OS_FS_ERR_PATH_INVALID

```
#define OS_FS_ERR_PATH_INVALID (-108)
```

FS path invalid.

Definition at line 131 of file osapi-error.h.

10.59.2.26 OS_FS_ERR_PATH_TOO_LONG

```
#define OS_FS_ERR_PATH_TOO_LONG (-103)
```

FS path too long.

Definition at line 127 of file osapi-error.h.

10.59.2.27 OS_INVALID_INT_NUM

```
#define OS_INVALID_INT_NUM (-5)
```

Invalid Interrupt number.

Definition at line 87 of file osapi-error.h.

10.59.2.28 OS_INVALID_POINTER

```
#define OS_INVALID_POINTER (-2)
```

Invalid pointer.

Definition at line 84 of file osapi-error.h.

10.59.2.29 OS_INVALID_SEM_VALUE

```
#define OS_INVALID_SEM_VALUE (-20)
```

Invalid semaphore value.

Definition at line 102 of file osapi-error.h.

10.59.2.30 OS_QUEUE_EMPTY

```
#define OS_QUEUE_EMPTY (-8)
```

Queue empty.

Definition at line 90 of file osapi-error.h.

10.59.2.31 OS_QUEUE_FULL

```
#define OS_QUEUE_FULL (-9)
```

Queue full.

Definition at line 91 of file osapi-error.h.

10.59.2.32 OS_QUEUE_ID_ERROR

```
#define OS_QUEUE_ID_ERROR (-12)
```

Queue ID error.

Definition at line 94 of file osapi-error.h.

10.59.2.33 OS_QUEUE_INVALID_SIZE

```
#define OS_QUEUE_INVALID_SIZE (-11)
```

Queue invalid size.

Definition at line 93 of file osapi-error.h.

10.59.2.34 OS_QUEUE_TIMEOUT

```
#define OS_QUEUE_TIMEOUT (-10)
```

Queue timeout.

Definition at line 92 of file osapi-error.h.

10.59.2.35 OS_SEM_FAILURE

```
#define OS_SEM_FAILURE (-6)
```

Semaphore failure.

Definition at line 88 of file osapi-error.h.

10.59.2.36 OS_SEM_TIMEOUT

```
#define OS_SEM_TIMEOUT (-7)
```

Semaphore timeout.

Definition at line 89 of file osapi-error.h.

10.59.2.37 OS_SUCCESS

```
#define OS_SUCCESS (0)
```

Successful execution.

Definition at line 82 of file osapi-error.h.

10.59.2.38 OS_TIMER_ERR_INTERNAL

```
#define OS_TIMER_ERR_INTERNAL (-32)
```

Timer internal error.

Definition at line 108 of file osapi-error.h.

10.59.2.39 OS_TIMER_ERR_INVALID_ARGS

```
#define OS_TIMER_ERR_INVALID_ARGS (-29)
```

Timer invalid arguments.

Definition at line 105 of file osapi-error.h.

10.59.2.40 OS_TIMER_ERR_TIMER_ID

```
#define OS_TIMER_ERR_TIMER_ID (-30)
```

Timer ID error.

Definition at line 106 of file osapi-error.h.

10.59.2.41 OS_TIMER_ERR_UNAVAILABLE

```
#define OS_TIMER_ERR_UNAVAILABLE (-31)
```

Timer unavailable.

Definition at line 107 of file osapi-error.h.

10.60 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.

10.60.1 Detailed Description

10.60.2 Function Documentation

10.60.2.1 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

Referenced by `OS_StatusToInteger()`.

10.60.2.2 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	<i>Status</i>	Execution status, see OSAL Return Code Defines
----	---------------	--

Returns

Same status value converted to the "long" data type

Definition at line 150 of file osapi-error.h.

References OS_GetErrorName().

Here is the call graph for this function:



10.61 OSAL File Access Option Defines

Macros

- #define OS_READ_ONLY 0
- #define OS_WRITE_ONLY 1
- #define OS_READ_WRITE 2

10.61.1 Detailed Description

10.61.2 Macro Definition Documentation

10.61.2.1 OS_READ_ONLY

```
#define OS_READ_ONLY 0
```

Read only file access

Definition at line 35 of file osapi-file.h.

10.61.2.2 OS_READ_WRITE

```
#define OS_READ_WRITE 2
```

Read write file access

Definition at line 37 of file osapi-file.h.

10.61.2.3 OS_WRITE_ONLY

```
#define OS_WRITE_ONLY 1
```

Write only file access

Definition at line 36 of file osapi-file.h.

10.62 OSAL Reference Point For Seek Offset Defines

Macros

- #define OS_SEEK_SET 0
- #define OS_SEEK_CUR 1
- #define OS_SEEK_END 2

10.62.1 Detailed Description

10.62.2 Macro Definition Documentation

10.62.2.1 OS_SEEK_CUR

```
#define OS_SEEK_CUR 1
```

Seek offset current

Definition at line 44 of file osapi-file.h.

10.62.2.2 OS_SEEK_END

```
#define OS_SEEK_END 2
```

Seek offset end

Definition at line 45 of file osapi-file.h.

10.62.2.3 OS_SEEK_SET

```
#define OS_SEEK_SET 0
```

Seek offset set

Definition at line 43 of file osapi-file.h.

10.63 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_TimedRead (osal_id_t filedes, void *buffer, size_t nbytes, int32 timeout)`
File/Stream input read 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.

10.63.1 Detailed Description

10.63.2 Function Documentation

10.63.2.1 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

10.63.2.2 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)

10.63.2.3 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)

10.63.2.4 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

10.63.2.5 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

10.63.2.6 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

10.63.2.7 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

10.63.2.8 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)

10.63.2.9 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)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	if the file could not be renamed.
OS_INVALID_POINTER	if src or dest are NULL
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_PATH_TOO_LONG	if the paths given are too long to be stored locally
OS_FS_ERR_NAME_TOO_LONG	if the dest name is too long to be stored locally

10.63.2.10 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

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	if the command was not executed properly
OS_INVALID_POINTER	if pointer argument was NULL
OS_ERR_NO_FREE_IDS	if all available file handles are in use
OS_FS_ERR_NAME_TOO_LONG	if the filename portion of the path exceeds OS_MAX_FILE_NAME
OS_FS_ERR_PATH_INVALID	if the path argument is not valid
OS_FS_ERR_PATH_TOO_LONG	if the path argument exceeds OS_MAX_PATH_LEN

10.63.2.11 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

10.63.2.12 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

10.63.2.13 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 or new 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

10.63.2.14 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

<i>in</i>	<i>path</i>	The file to operate on (must not be null)
<i>out</i>	<i>filestats</i>	Buffer to store file 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 path or filestats is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path is too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the name of the file is too long to be stored
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_ERROR</i>	if the OS call failed

10.63.2.15 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 (OS_PEND = forever)

Returns

Byte count on success 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 at end of file/stream data

10.63.2.16 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 (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

10.63.2.17 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

<i>OS_INVALID_POINTER</i>	if buffer is NULL
<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 file/stream cannot accept any more data

10.64 OSAL File System Level APIs

Functions

- `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.
- `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 *filesystem_info)`
Returns information about the file system.

10.64.1 Detailed Description

10.64.2 Function Documentation

10.64.2.1 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	<i>name</i>	The device/path to operate on (must not be null)
in	<i>repair</i>	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)

10.64.2.2 OS_FileSysAddFixedMap()

```
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.

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>filesys_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 phys_path is too long
<i>OS_ERR_NAME_TOO_LONG</i>	if the phys_path basename (filesystem name) is too long
<i>OS_INVALID_POINTER</i>	if any argument is NULL

10.64.2.3 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

<i>in</i>	<i>name</i>	The device/path to operate on (must not be null)
<i>out</i>	<i>statbuf</i>	Output structure to populate (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 or statbuf is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the name is too long
<i>OS_ERROR</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.64.2.4 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if either parameter is NULL
<i>OS_ERR_NAME_NOT_FOUND</i>	if the MountPoint is not mounted in OSAL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the MountPoint is too long

10.64.2.5 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

out	<i>filesys_info</i>	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

10.64.2.6 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

in	<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
in	<code>devname</code>	The underlying kernel device to use, if applicable. (must not be null)
in	<code>volname</code>	The name of the volume (see note) (must not be null)
in	<code>blocksize</code>	The size of a single block on the drive
in	<code>numblocks</code>	The number of blocks to allocate for the drive

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>devname</code> or <code>volname</code> are NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_FS_ERR_DEVICE_NOT_FREE</code>	if the volume table is full
<code>OS_FS_ERR_DRIVE_NOT_CREATED</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.64.2.7 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

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if devname or volname is NULL
OS_FS_ERR_PATH_TOO_LONG	if the overall devname or volname is too long
OS_FS_ERR_DEVICE_NOT_FREE	if the volume table is full
OS_FS_ERR_DRIVE_NOT_CREATED	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.64.2.8 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. <i>devname</i> 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)

10.64.2.9 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.
----------------------------	-----------------------

Return values

<i>OS_INVALID_POINTER</i>	if devname is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the devname is too long
<i>OS_ERR_NAME_NOT_FOUND</i>	if the devname does not exist in OSAL
<i>OS_ERROR</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.64.2.10 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

10.64.2.11 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

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if name is NULL
OS_FS_ERR_PATH_TOO_LONG	if the absolute path given is too long
OS_ERR_NAME_NOT_FOUND	if the mountpoint is not mounted in OSAL
OS_ERROR	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.65 OSAL Heap APIs

Functions

- `int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)`

Return current info on the heap.

10.65.1 Detailed Description

10.65.2 Function Documentation

10.65.2.1 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 <code>heap_prop</code> argument is NULL

10.66 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_USER** 0x10
Object user type.

10.66.1 Detailed Description

10.66.2 Macro Definition Documentation

10.66.2.1 **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.

10.66.2.2 OS_OBJECT_TYPE_OS_CONSOLE

```
#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C
```

Object console type.

Definition at line 50 of file osapi-idmap.h.

10.66.2.3 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.

10.66.2.4 OS_OBJECT_TYPE_OS_DIR

```
#define OS_OBJECT_TYPE_OS_DIR 0x07
```

Object directory type.

Definition at line 45 of file osapi-idmap.h.

10.66.2.5 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.

10.66.2.6 OS_OBJECT_TYPE_OS_MODULE

```
#define OS_OBJECT_TYPE_OS_MODULE 0x0A
```

Object module type.

Definition at line 48 of file osapi-idmap.h.

10.66.2.7 OS_OBJECT_TYPE_OS_MUTEX

```
#define OS_OBJECT_TYPE_OS_MUTEX 0x05
```

Object mutex type.

Definition at line 43 of file osapi-idmap.h.

10.66.2.8 OS_OBJECT_TYPE_OS_QUEUE

```
#define OS_OBJECT_TYPE_OS_QUEUE 0x02
```

Object queue type.

Definition at line 40 of file osapi-idmap.h.

10.66.2.9 OS_OBJECT_TYPE_OS_STREAM

```
#define OS_OBJECT_TYPE_OS_STREAM 0x06
```

Object stream type.

Definition at line 44 of file osapi-idmap.h.

10.66.2.10 OS_OBJECT_TYPE_OS_TASK

```
#define OS_OBJECT_TYPE_OS_TASK 0x01
```

Object task type.

Definition at line 39 of file osapi-idmap.h.

10.66.2.11 OS_OBJECT_TYPE_OS_TIMEBASE

```
#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08
```

Object timebase type.

Definition at line 46 of file osapi-idmap.h.

10.66.2.12 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.

10.66.2.13 OS_OBJECT_TYPE_UNDEFINED

```
#define OS_OBJECT_TYPE_UNDEFINED 0x00
```

Object type undefined.

Definition at line 38 of file osapi-idmap.h.

10.66.2.14 OS_OBJECT_TYPE_USER

```
#define OS_OBJECT_TYPE_USER 0x10
```

Object user type.

Definition at line 51 of file osapi-idmap.h.

10.67 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

10.67.1 Detailed Description

10.67.2 Function Documentation

10.67.2.1 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	<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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the object_id argument is not valid
<i>OS_INVALID_POINTER</i>	if the ArrayIndex is NULL

Referenced by `OS_ObjectIdDefined()`.

10.67.2.2 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 <code>OS_OBJECT_CREATOR_ANY</code> 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)

Referenced by `OS_ObjectIdDefined()`.

10.67.2.3 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)

Referenced by OS_ObjectIdDefined().

10.67.2.4 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

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the passed-in ID is not a valid OSAL ID
OS_INVALID_POINTER	if the passed-in buffer is invalid
OS_ERR_NAME_TOO_LONG	if the name will not fit in the buffer provided

Referenced by OS_ObjectIdDefined().

10.67.2.5 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

Referenced by OS_ObjectIdDefined().

10.67.2.6 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 147 of file osapi-idmap.h.

References OS_ConvertToArrayIndex(), OS_ForEachObject(), OS_ForEachObjectType(), OS_GetResourceName(), OS_IdentifyObject(), OS_ObjectIdToArrayIndex(), and OS_ObjectIdToInteger().

10.67.2.7 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 126 of file osapi-idmap.h.

References OS_ObjectIdToInteger().

10.67.2.8 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 101 of file osapi-idmap.h.

10.67.2.9 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

Referenced by [OS_ObjectIdDefined\(\)](#).

10.67.2.10 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 79 of file osapi-idmap.h.

Referenced by OS_ObjectIdDefined(), and OS_ObjectIdEqual().

10.68 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.

10.68.1 Detailed Description

10.68.2 Function Documentation

10.68.2.1 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

Return values

<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)

10.68.2.2 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

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if one of the parameters is NULL
<code>OS_ERR_NO_FREE_IDS</code>	if the module table is full
<code>OS_ERR_NAME_TAKEN</code>	if the name is in use
<code>OS_ERR_NAME_TOO_LONG</code>	if the module_name is too long
<code>OS_FS_ERR_PATH_INVALID</code>	if the filename argument is not valid
<code>OS_ERROR</code>	if an other/unspecified error occurs (return value only verified in coverage test)

10.68.2.3 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.
OS_ERROR	if the symbol could not be found
OS_INVALID_POINTER	if one of the pointers passed in are NULL

10.68.2.4 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)

10.68.2.5 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

10.68.2.6 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)

10.69 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.

10.69.1 Detailed Description

10.69.2 Function Documentation

10.69.2.1 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if sem_id or sem_name 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 there are no more free mutex Ids
<i>OS_ERR_NAME_TAKEN</i>	if there is already a mutex with the same name
<i>OS_SEM_FAILURE</i>	if the OS call failed (return value only verified in coverage test)

10.69.2.2 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	<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 mutex
<i>OS_SEM_FAILURE</i>	if an unspecified error occurs (return value only verified in coverage test)

10.69.2.3 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	<i>sem_id</i>	will be set to the ID of the existing resource
in	<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

10.69.2.4 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

10.69.2.5 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	<code>sem_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 mutex
<code>OS_SEM_FAILURE</code>	if an unspecified error occurs (return value only verified in coverage test)

10.69.2.6 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	<code>sem_id</code>	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)

10.70 OSAL Network ID APIs

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.

10.70.1 Detailed Description

Provides some basic methods to query a network host name and ID

10.70.2 Function Documentation

10.70.2.1 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

out	<i>host_name</i>	Buffer to hold name information (must not be null)
in	<i>name_len</i>	Maximum length of host name buffer (must not be zero)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_SIZE	if the name_len is zero
OS_INVALID_POINTER	if the host_name is NULL

10.70.2.2 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.

10.71 OSAL Printf APIs

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.

10.71.1 Detailed Description

10.71.2 Function Documentation

10.71.2.1 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	string	Format string, followed by additional arguments
----	--------	---

Referenced by HS_ProcessMain().

10.71.2.2 OS_printf_disable()

```
void void OS_printf_disable (
    void )
```

This function disables the output from OS_printf.

10.71.2.3 OS_printf_enable()

```
void OS_printf_enable (
    void )
```

This function enables the output from OS_printf.

10.72 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.

10.72.1 Detailed Description

10.72.2 Function Documentation

10.72.2.1 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

out	<code>queue_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>queue_name</code>	the name of the new resource to create (must not be null)
in	<code>queue_depth</code>	the maximum depth of the queue
in	<code>data_size</code>	the size of each entry in the queue (must not be zero)
in	<code>flags</code>	options for the queue (reserved for future use, pass as 0)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if a pointer passed in is NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than OS_MAX_API_NAME
<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

10.72.2.2 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)

10.72.2.3 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
out	<i>data</i>	The buffer to store the received message (must not be null)
in	<i>size</i>	The size of the data buffer (must not be zero)
out	<i>size_copied</i>	Set to the actual size of the message (must not be null)
in	<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)

10.72.2.4 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

out	<i>queue_id</i>	will be set to the ID of the existing resource
in	<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

10.72.2.5 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

10.72.2.6 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)

10.73 OSAL Select APIs

Functions

- `int32 OS_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)`
Wait for events across multiple file handles.
- `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_SelectFdIsSet (const OS_FdSet *Set, osal_id_t objid)`
Check if an FdSet structure contains a given ID.

10.73.1 Detailed Description

10.73.2 Function Documentation

10.73.2.1 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

in, out	<code>Set</code>	Pointer to <code>OS_FdSet</code> object to operate on (must not be null)
in	<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

10.73.2.2 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

10.73.2.3 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

10.73.2.4 OS_SelectFdZero()

```
int32 OS_SelectFdZero (
    OS_FdSet * Set )
```

Clear a FdSet structure.

After this call the set will contain no OSAL IDs

Parameters

<i>out</i>	<i>Set</i>	Pointer to OS_FdSet object to clear (must not be null)
------------	------------	--

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL

10.73.2.5 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.

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

<i>in, out</i>	<i>ReadSet</i>	Set of handles to check/wait to become readable
<i>in, out</i>	<i>WriteSet</i>	Set of handles to check/wait to become writable
<i>in</i>	<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)

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

OS_SUCCESS	If any handle in the ReadSet or WriteSet is readable or writable, respectively
OS_ERROR_TIMEOUT	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
OS_ERR_OPERATION_NOT_SUPPORTED	if a specified handle does not support select
OS_ERR_INVALID_ID	if no valid handles were contained in the ReadSet/WriteSet

10.73.2.6 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.

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)

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

10.74 OSAL Shell APIs

Functions

- `int32 OS_ShellOutputToFile (const char *Cmd, osal_id_t filedes)`

Executes the command and sends output to a file.

10.74.1 Detailed Description

10.74.2 Function Documentation

10.74.2.1 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

10.75 OSAL Socket Address APIs

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.

10.75.1 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.

10.75.2 Function Documentation

10.75.2.1 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

<i>out</i>	<i>Addr</i>	The address buffer to initialize (must not be null)
<i>in</i>	<i>string</i>	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

10.75.2.2 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

10.75.2.3 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

out	<i>Addr</i>	The address buffer to initialize (must not be null)
in	<i>Domain</i>	The address family

Returns

Execution status, see [OSAL Return Code Defines](#)

Return values

<i>OS_SUCCESS</i>	Successful execution.
<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

10.75.2.4 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

out	<i>Addr</i>	The network address buffer (must not be null)
in	<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

10.75.2.5 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

out	<i>buffer</i>	Buffer to hold the output string (must not be null)
in	<i>buflen</i>	Maximum length of the output string (must not be zero)
in	<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

10.76 OSAL Socket Management APIs

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.
- `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_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_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.

10.76.1 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.

10.76.2 Function Documentation

10.76.2.1 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.

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

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_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_ERR_INCORRECT_OBJ_STATE</i>	if the socket is not bound or already connected

10.76.2.2 OS_SocketBind()

```
int32 OS_SocketBind (
    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.

If the socket is connectionless, then it only binds to the local address.

If the socket is connection-oriented (stream), then this will also put the socket into a listening state for incoming connections at the local address.

Parameters

in	<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

10.76.2.3 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).

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 <i>OS_PEND</i> to wait forever

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

10.76.2.4 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

out	<i>sock_id</i>	Buffer to hold result (must not be null)
in	<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 OS_MAX_API_NAME
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

10.76.2.5 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

in	<i>sock_id</i>	The socket ID
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

10.76.2.6 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)
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

10.76.2.7 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.

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

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 <i>sock_id</i> parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket

10.76.2.8 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

<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

10.76.2.9 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	<i>sock_id</i>	The socket ID
in	<i>Mode</i>	Whether to shutdown reading, writing, or both.

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_INVALID_ARGUMENT</i>	if the Mode argument is not one of the valid options
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is not connected

10.77 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.

10.77.1 Detailed Description

10.77.2 Function Documentation

10.77.2.1 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

out	<i>task_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
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)

10.77.2.2 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 HS_AppMain(), HS_MonitorApplications(), and HS_MonitorEvent().

10.77.2.3 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the ID given to it is invalid
<i>OS_ERROR</i>	if the OS delete call fails (return value only verified in coverage test)

10.77.2.4 OS_TaskExit()

```
void OS_TaskExit (
    void )
```

Exits the calling task.

The calling thread is terminated. This function does not return.

10.77.2.5 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

<i>OS_SUCCESS</i>	Successful execution. (return value only verified in coverage test)
<i>OS_INVALID_POINTER</i>	if a pointer argument is NULL

10.77.2.6 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)

10.77.2.7 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 <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name wasn't found in the table

10.77.2.8 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

10.77.2.9 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

in	<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
--------------------------	--

10.77.2.10 OS_TaskSetPriority()

```
int32 OS_TaskSetPriority (
    osal_id_t task_id,
    osal_priority_t new_priority )
```

Sets the given task to a new priority.

Parameters

in	<i>task_id</i>	The object ID to operate on
in	<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
<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)

10.78 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.

10.78.1 Detailed Description

10.78.2 Function Documentation

10.78.2.1 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

out	<i>timebase_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<i>timebase_name</i>	The name of the time base (must not be null)
in	<i>external_sync</i>	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

10.78.2.2 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

10.78.2.3 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

10.78.2.4 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 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

10.78.2.5 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

10.78.2.6 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

10.79 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.

10.79.1 Detailed Description

10.79.2 Function Documentation

10.79.2.1 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

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)
in	<i>timebase_id</i>	The time base resource to use as a reference
in	<i>callback_ptr</i>	Application-provided function to invoke (must not be null)
in	<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)

10.79.2.2 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

<code>out</code>	<code>timer_id</code>	Will be set to the non-zero resource ID of the timer object (must not be null)
<code>in</code>	<code>timer_name</code>	Name of the timer object (must not be null)
<code>out</code>	<code>clock_accuracy</code>	Expected precision of the timer, in microseconds. This is the underlying tick value rounded to the nearest microsecond integer. (must not be null)
<code>in</code>	<code>callback_ptr</code>	The function pointer of the timer callback (must not 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_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)

10.79.2.3 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

10.79.2.4 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.
-------------------	-----------------------

Return values

<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 <i>OS_MAX_API_NAME</i>
<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

10.79.2.5 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.

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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timer
<i>OS_INVALID_POINTER</i>	if the timer_prop pointer is null
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

10.79.2.6 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 75 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 78 of file ccsds_hdr.h.

11.1.2.2 SystemId

```
uint8 CCSDS_ExtendedHeader::SystemId[2]
```

system qualifier

Definition at line 85 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 66 of file ccsds_hdr.h.

11.2.2.2 Sequence

```
uint8 CCSDS_PrimaryHeader::Sequence[2]
```

packet sequence word

Definition at line 61 of file ccsds_hdr.h.

11.2.2.3 StreamId

`uint8 CCSDS_PrimaryHeader::StreamId[2]`

packet identifier word (stream ID)

Definition at line 54 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 CFE_ES_AppInfo Struct Reference

Application Information.

```
#include <cfe_es_extern_typedefs.h>
```

Data Fields

- **CFE_Resourceld_t Resourceld**
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.

11.3.1 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 442 of file cfe_es_extern_typedefs.h.

11.3.2 Field Documentation

11.3.2.1 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 458 of file cfe_es_extern_typedefs.h.

11.3.2.2 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 468 of file cfe_es_extern_typedefs.h.

11.3.2.3 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 470 of file cfe_es_extern_typedefs.h.

11.3.2.4 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 460 of file cfe_es_extern_typedefs.h.

11.3.2.5 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 462 of file cfe_es_extern_typedefs.h.

11.3.2.6 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 464 of file cfe_es_extern_typedefs.h.

11.3.2.7 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 466 of file cfe_es_extern_typedefs.h.

11.3.2.8 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 451 of file cfe_es_extern_typedefs.h.

11.3.2.9 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 474 of file cfe_es_extern_typedefs.h.

11.3.2.10 ExecutionCounter

`uint32 CFE_ES_AppInfo::ExecutionCounter`

The Application's Main Task Execution Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ExecutionCtr

Definition at line 481 of file cfe_es_extern_typedefs.h.

Referenced by HS_MonitorApplications().

11.3.2.11 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 453 of file cfe_es_extern_typedefs.h.

11.3.2.12 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 479 of file cfe_es_extern_typedefs.h.

11.3.2.13 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 483 of file cfe_es_extern_typedefs.h.

11.3.2.14 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 449 of file cfe_es_extern_typedefs.h.

11.3.2.15 NumOfChildTasks

```
uint32 CFE_ES_AppInfo::NumOfChildTasks
```

Number of Child tasks for an App.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ChildTasks

Definition at line 485 of file cfe_es_extern_typedefs.h.

11.3.2.16 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 477 of file cfe_es_extern_typedefs.h.

11.3.2.17 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 444 of file cfe_es_extern_typedefs.h.

11.3.2.18 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 456 of file `cfe_es_extern_typedefs.h`.

11.3.2.19 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 472 of file `cfe_es_extern_typedefs.h`.

11.3.2.20 Type

`uint32 CFE_ES_AppInfo::Type`

The type of App: CORE or EXTERNAL.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_AppType

Definition at line 446 of file `cfe_es_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h`

11.4 CFE_ES_AppNameCmd Struct Reference

Generic application name command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_AppNameCmd_Payload_t](#) Payload
Command payload.

11.4.1 Detailed Description

Generic application name command.

Definition at line 1198 of file cfe_es_msg.h.

11.4.2 Field Documentation

11.4.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_AppNameCmd::CommandHeader

Command header.

Definition at line 1200 of file cfe_es_msg.h.

11.4.2.2 Payload

[CFE_ES_AppNameCmd_Payload_t](#) CFE_ES_AppNameCmd::Payload

Command payload.

Definition at line 1201 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.5 CFE_ES_AppNameCmd_Payload Struct Reference

Generic application name command payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char [Application \[CFE_MISSION_MAX_API_LEN\]](#)
ASCII text string containing Application or Library Name.

11.5.1 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 1190 of file cfe_es_msg.h.

11.5.2 Field Documentation

11.5.2.1 Application

```
char CFE_ES_AppNameCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]
```

ASCII text string containing Application or Library Name.

Definition at line 1192 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.6 CFE_ES_AppReloadCmd_Payload Struct Reference

Reload Application Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char [Application \[CFE_MISSION_MAX_API_LEN\]](#)
ASCII text string containing Application Name.
- char [AppFileName \[CFE_MISSION_MAX_PATH_LEN\]](#)
Full path and filename of Application's executable image.

11.6.1 Detailed Description

Reload Application Command Payload.

For command details, see [CFE_ES_RELOAD_APP_CC](#)

Definition at line 1219 of file cfe_es_msg.h.

11.6.2 Field Documentation

11.6.2.1 AppFileName

```
char CFE_ES_AppReloadCmd_Payload::AppFileName[CFE\_MISSION\_MAX\_PATH\_LEN]
```

Full path and filename of Application's executable image.

Definition at line 1222 of file cfe_es_msg.h.

11.6.2.2 Application

```
char CFE_ES_AppReloadCmd_Payload::Application[CFE\_MISSION\_MAX\_API\_LEN]
```

ASCII text string containing Application Name.

Definition at line 1221 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.7 CFE_ES_BlockStats Struct Reference

Block statistics.

```
#include <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.

11.7.1 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 540 of file `cfe_es_extern_typedefs.h`.

11.7.2 Field Documentation

11.7.2.1 BlockSize

`CFE_ES_MemOffset_t CFE_ES_BlockStats::BlockSize`

Number of bytes in each of these blocks.

Definition at line 542 of file `cfe_es_extern_typedefs.h`.

11.7.2.2 NumCreated

`uint32 CFE_ES_BlockStats::NumCreated`

Number of Memory Blocks of this size created.

Definition at line 543 of file `cfe_es_extern_typedefs.h`.

11.7.2.3 NumFree

`uint32 CFE_ES_BlockStats::NumFree`

Number of Memory Blocks of this size that are free.

Definition at line 544 of file `cfe_es_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h`

11.8 CFE_ES_CDSRegDumpRec Struct Reference

CDS Register Dump Record.

```
#include <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.

11.8.1 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 525 of file `cfe_es_extern_typedefs.h`.

11.8.2 Field Documentation

11.8.2.1 ByteAlignSpare

`uint8 CFE_ES_CDSRegDumpRec::ByteAlignSpare[3]`

Spare bytes to ensure structure size is multiple of 4 bytes.

Definition at line 531 of file `cfe_es_extern_typedefs.h`.

11.8.2.2 Handle

`CFE_ES_CDSHandle_t CFE_ES_CDSRegDumpRec::Handle`

Handle of CDS.

Definition at line 527 of file `cfe_es_extern_typedefs.h`.

11.8.2.3 Name

```
char CFE_ES_CDSRegDumpRec::Name[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]
```

Processor Unique Name of CDS.

Definition at line 530 of file cfe_es_extern_typedefs.h.

11.8.2.4 Size

```
CFE_ES_MemOffset_t CFE_ES_CDSRegDumpRec::Size
```

Size, in bytes, of the CDS memory block.

Definition at line 528 of file cfe_es_extern_typedefs.h.

11.8.2.5 Table

```
bool CFE_ES_CDSRegDumpRec::Table
```

Flag that indicates whether CDS contains a Critical Table.

Definition at line 529 of file cfe_es_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h

11.9 CFE_ES_DeleteCDSCmd Struct Reference

Delete Critical Data Store Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_DeleteCDSCmd_Payload_t](#) Payload
Command payload.

11.9.1 Detailed Description

Delete Critical Data Store Command.

Definition at line 1272 of file cfe_es_msg.h.

11.9.2 Field Documentation

11.9.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_ES_DeleteCDSCmd::CommandHeader`

Command header.

Definition at line 1274 of file cfe_es_msg.h.

11.9.2.2 Payload

`CFE_ES_DeleteCDSCmd_Payload_t` `CFE_ES_DeleteCDSCmd::Payload`

Command payload.

Definition at line 1275 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.10 CFE_ES_DeleteCDSCmd_Payload Struct Reference

Delete Critical Data Store Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char `CdsName [CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`
ASCII text string containing name of CDS to delete.

11.10.1 Detailed Description

Delete Critical Data Store Command Payload.

For command details, see [CFE_ES_DELETE_CDS_CC](#)

Definition at line 1262 of file cfe_es_msg.h.

11.10.2 Field Documentation

11.10.2.1 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 1265 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.11 CFE_ES_DumpCDSRegistryCmd Struct Reference

Dump CDS Registry Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_DumpCDSRegistryCmd_Payload_t](#) Payload
Command payload.

11.11.1 Detailed Description

Dump CDS Registry Command.

Definition at line 1400 of file cfe_es_msg.h.

11.11.2 Field Documentation

11.11.2.1 CommandHeader

[`CFE_MSG_CommandHeader_t`](#) `CFE_ES_DumpCDSRegistryCmd::CommandHeader`

Command header.

Definition at line 1402 of file `cfe_es_msg.h`.

11.11.2.2 Payload

[`CFE_ES_DumpCDSRegistryCmd_Payload_t`](#) `CFE_ES_DumpCDSRegistryCmd::Payload`

Command payload.

Definition at line 1403 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.12 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference

Dump CDS Registry Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `char DumpFilename [CFE_MISSION_MAX_PATH_LEN]`
ASCII text string of full path and filename of file CDS Registry is to be written.

11.12.1 Detailed Description

Dump CDS Registry Command Payload.

For command details, see [CFE_ES_DUMP_CDS_REGISTRY_CC](#)

Definition at line 1391 of file `cfe_es_msg.h`.

11.12.2 Field Documentation

11.12.2.1 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 1393 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/cfe_es_msg.h

11.13 CFE_ES_FileNameCmd Struct Reference

Generic file name command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_FileNameCmd_Payload_t](#) Payload
Command payload.

11.13.1 Detailed Description

Generic file name command.

Definition at line 1115 of file cfe_es_msg.h.

11.13.2 Field Documentation

11.13.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_FileNameCmd::CommandHeader

Command header.

Definition at line 1117 of file cfe_es_msg.h.

11.13.2.2 Payload

`CFE_ES_FileNameCmd_Payload_t` `CFE_ES_FileNameCmd::Payload`

Command payload.

Definition at line 1118 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.14 CFE_ES_FileNameCmd_Payload Struct Reference

Generic file name command payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `char FileName [CFE_MISSION_MAX_PATH_LEN]`

ASCII text string containing full path and filename of file in which Application data is to be dumped.

11.14.1 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_SYSLOG_CC](#), and [CFE_ES_WRITE_ER_LOG_CC](#)

Definition at line 1106 of file `cfe_es_msg.h`.

11.14.2 Field Documentation

11.14.2.1 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 1108 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.15 CFE_ES_HousekeepingTlm Struct Reference

```
#include <cfes_es_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_ES_HousekeepingTlm_Payload_t Payload](#)
Telemetry payload.

11.15.1 Detailed Description

Definition at line 1548 of file cfe_es_msg.h.

11.15.2 Field Documentation

11.15.2.1 Payload

[CFE_ES_HousekeepingTlm_Payload_t](#) CFE_ES_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 1551 of file cfe_es_msg.h.

11.15.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_ES_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 1550 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfes_es_msg.h](#)

11.16 CFE_ES_HousekeepingTlm_Payload Struct Reference

```
#include <cfes_es_msg.h>
```

Data Fields

- **uint8 CommandCounter**
The ES Application Command Counter.
- **uint8 CommandErrorCounter**
The ES Application Command Error Counter.
- **uint16 CFECOREChecksum**
Checksum of cFE Core Code.
- **uint8 CFEMajorVersion**
Major Version Number of cFE.
- **uint8 CFEMinorVersion**
Minor Version Number of cFE.
- **uint8 CFERevision**
Sub-Minor Version Number of cFE.
- **uint8 CFEMissionRevision**
Mission Version Number of cFE.
- **uint8 OSALMajorVersion**
OS Abstraction Layer Major Version Number.
- **uint8 OSALMinorVersion**
OS Abstraction Layer Minor Version Number.
- **uint8 OSALRevision**
OS Abstraction Layer Revision Number.
- **uint8 OSALMissionRevision**
OS Abstraction Layer MissionRevision Number.
- **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**

- **uint32 RegisteredTasks**
Number of Applications 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.

11.16.1 Detailed Description

Name Executive Services Housekeeping Packet

Definition at line 1451 of file cfe_es_msg.h.

11.16.2 Field Documentation

11.16.2.1 BootSource

`uint32 CFE_ES_HousekeepingTlm_Payload::BootSource`

Boot source (as provided from BSP)

Telemetry Mnemonic(s) \$sc_\$cpu_ES_BootSource

Definition at line 1517 of file cfe_es_msg.h.

11.16.2.2 CFECOREChecksum

`uint16 CFE_ES_HousekeepingTlm_Payload::CFECOREChecksum`

Checksum of cFE Core Code.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CKSUM

Definition at line 1458 of file cfe_es_msg.h.

11.16.2.3 CFEMajorVersion

`uint8 CFE_ES_HousekeepingTlm_Payload::CFEMajorVersion`

Major Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMAJORVER

Definition at line 1460 of file cfe_es_msg.h.

11.16.2.4 CFEMinorVersion

`uint8 CFE_ES_HousekeepingTlm_Payload::CFEMinorVersion`

Minor Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMINORVER

Definition at line 1462 of file cfe_es_msg.h.

11.16.2.5 CFEMissionRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::CFEMissionRevision
```

Mission Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEMISSIONREV

Definition at line 1466 of file cfe_es_msg.h.

11.16.2.6 CFERevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::CFERevision
```

Sub-Minor Version Number of cFE.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CFEREVISION

Definition at line 1464 of file cfe_es_msg.h.

11.16.2.7 CommandCounter

```
uint8 CFE_ES_HousekeepingTlm_Payload::CommandCounter
```

The ES Application Command Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CMDPC

Definition at line 1453 of file cfe_es_msg.h.

11.16.2.8 CommandErrorCounter

```
uint8 CFE_ES_HousekeepingTlm_Payload::CommandErrorCounter
```

The ES Application Command Error Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_CMDEC

Definition at line 1455 of file cfe_es_msg.h.

11.16.2.9 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 1497 of file cfe_es_msg.h.

11.16.2.10 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 1495 of file cfe_es_msg.h.

11.16.2.11 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 1542 of file cfe_es_msg.h.

11.16.2.12 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 1540 of file cfe_es_msg.h.

11.16.2.13 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 1544 of file cfe_es_msg.h.

11.16.2.14 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 1515 of file cfe_es_msg.h.

11.16.2.15 OSALMajorVersion

`uint8 CFE_ES_HousekeepingTlm_Payload::OSALMajorVersion`

OS Abstraction Layer Major Version Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_OSMajorVer

Definition at line 1468 of file cfe_es_msg.h.

11.16.2.16 OSALMinorVersion

`uint8 CFE_ES_HousekeepingTlm_Payload::OSALMinorVersion`

OS Abstraction Layer Minor Version Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_OsminorVer

Definition at line 1470 of file cfe_es_msg.h.

11.16.2.17 OSALMissionRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::OSALMissionRevision
```

OS Abstraction Layer MissionRevision Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_OSMISSIONREV

Definition at line 1474 of file cfe_es_msg.h.

11.16.2.18 OSALRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::OSALRevision
```

OS Abstraction Layer Revision Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_OSREVISION

Definition at line 1472 of file cfe_es_msg.h.

11.16.2.19 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 1535 of file cfe_es_msg.h.

11.16.2.20 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 1533 of file cfe_es_msg.h.

11.16.2.21 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 1531 of file cfe_es_msg.h.

11.16.2.22 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 1538 of file cfe_es_msg.h.

11.16.2.23 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 1526 of file cfe_es_msg.h.

11.16.2.24 PerfMode

```
uint32 CFE_ES_HousekeepingTlm_Payload::PerfMode
```

Current mode of Performance Analyzer.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfMode

Definition at line 1522 of file cfe_es_msg.h.

11.16.2.25 PerfState

```
uint32 CFE_ES_HousekeepingTlm_Payload::PerfState
```

Current state of Performance Analyzer.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfState

Definition at line 1520 of file cfe_es_msg.h.

11.16.2.26 PerfTriggerCount

```
uint32 CFE_ES_HousekeepingTlm_Payload::PerfTriggerCount
```

Number of Times Performance Analyzer has Triggered.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PerfTrigCnt

Definition at line 1524 of file cfe_es_msg.h.

11.16.2.27 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 1529 of file cfe_es_msg.h.

11.16.2.28 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 1513 of file cfe_es_msg.h.

11.16.2.29 PSPMajorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion
```

Platform Support Package Major Version Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PSPMAJORVER

Definition at line 1477 of file cfe_es_msg.h.

11.16.2.30 PSPMinorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion
```

Platform Support Package Minor Version Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PSPMINORVER

Definition at line 1479 of file cfe_es_msg.h.

11.16.2.31 PSPMissionRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision
```

Platform Support Package MissionRevision Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PSPMISSIONREV

Definition at line 1483 of file cfe_es_msg.h.

11.16.2.32 PSPRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload::PSPRevision
```

Platform Support Package Revision Number.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_PSPREVISION

Definition at line 1481 of file cfe_es_msg.h.

11.16.2.33 RegisteredCoreApps

```
uint32 CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps
```

Number of Applications registered with ES.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_RegCoreApps

Definition at line 1500 of file cfe_es_msg.h.

11.16.2.34 RegisteredExternalApps

```
uint32 CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps
```

Number of Applications registered with ES.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_RegExtApps

Definition at line 1502 of file cfe_es_msg.h.

11.16.2.35 RegisteredLibs

```
uint32 CFE_ES_HousekeepingTlm_Payload::RegisteredLibs
```

Number of Libraries registered with ES.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_RegLibs

Definition at line 1506 of file cfe_es_msg.h.

11.16.2.36 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 1504 of file cfe_es_msg.h.

11.16.2.37 ResetSubtype

`uint32 CFE_ES_HousekeepingTlm_Payload::ResetSubtype`

Reset Sub Type.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ResetSubtype

Definition at line 1511 of file cfe_es_msg.h.

11.16.2.38 ResetType

`uint32 CFE_ES_HousekeepingTlm_Payload::ResetType`

Reset type (PROCESSOR or POWERON)

Telemetry Mnemonic(s) \$sc_\$cpu_ES_ResetType

Definition at line 1509 of file cfe_es_msg.h.

11.16.2.39 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 1486 of file cfe_es_msg.h.

11.16.2.40 SysLogEntries

`uint32 CFE_ES_HousekeepingTlm_Payload::SysLogEntries`

Number of entries in the system log.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_SYSLOGENTRIES

Definition at line 1490 of file cfe_es_msg.h.

11.16.2.41 SysLogMode

`uint32 CFE_ES_HousekeepingTlm_Payload::SysLogMode`

Write/Overwrite Mode.

Telemetry Mnemonic(s) \$sc_\$cpu_ES_SYSLOGMODE

Definition at line 1492 of file cfe_es_msg.h.

11.16.2.42 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 1488 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/cfe_es_msg.h

11.17 CFE_ES_MemPoolStats Struct Reference

Memory Pool Statistics.

```
#include <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.

11.17.1 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 555 of file cfe_es_extern_typedefs.h.

11.17.2 Field Documentation

11.17.2.1 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 565 of file cfe_es_extern_typedefs.h.

11.17.2.2 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 561 of file cfe_es_extern_typedefs.h.

11.17.2.3 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 559 of file cfe_es_extern_typedefs.h.

11.17.2.4 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 563 of file cfe_es_extern_typedefs.h.

11.17.2.5 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 557 of file cfe_es_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h

11.18 CFE_ES_MemStatsTlm Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_ES_PoolStatsTlm_Payload_t Payload`
Telemetry payload.

11.18.1 Detailed Description

Definition at line 1440 of file `cfe_es_msg.h`.

11.18.2 Field Documentation

11.18.2.1 Payload

`CFE_ES_PoolStatsTlm_Payload_t CFE_ES_MemStatsTlm::Payload`

Telemetry payload.

Definition at line 1443 of file `cfe_es_msg.h`.

11.18.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t CFE_ES_MemStatsTlm::TelemetryHeader`

Telemetry header.

Definition at line 1442 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.19 CFE_ES_NoArgsCmd Struct Reference

Generic "no arguments" command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.

11.19.1 Detailed Description

Generic "no arguments" command.

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see [CFE_ES_NOOP_CC](#))
3. The Reset Counters Command (For details, see [CFE_ES_RESET_COUNTERS_CC](#))

Definition at line 1059 of file cfe_es_msg.h.

11.19.2 Field Documentation

11.19.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_NoArgsCmd::CommandHeader

Command header.

Definition at line 1061 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.20 CFE_ES_OneAppTlm Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_ES_OneAppTlm_Payload_t](#) Payload
Telemetry payload.

11.20.1 Detailed Description

Definition at line 1424 of file cfe_es_msg.h.

11.20.2 Field Documentation

11.20.2.1 Payload

[CFE_ES_OneAppTlm_Payload_t](#) CFE_ES_OneAppTlm::Payload

Telemetry payload.

Definition at line 1427 of file cfe_es_msg.h.

11.20.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_ES_OneAppTlm::TelemetryHeader

Telemetry header.

Definition at line 1426 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.21 CFE_ES_OneAppTlm_Payload Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_ES_AppInfo_t AppInfo](#)

For more information, see [CFE_ES_AppInfo_t](#).

11.21.1 Detailed Description

Name Single Application Information Packet

Definition at line 1418 of file cfe_es_msg.h.

11.21.2 Field Documentation

11.21.2.1 ApplInfo

`CFE_ES_AppInfo_t` `CFE_ES_OneAppTlm_Payload::AppInfo`

For more information, see [CFE_ES_AppInfo_t](#).

Definition at line 1420 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.22 CFE_ES_OverWriteSysLogCmd Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_ES_OverWriteSysLogCmd_Payload_t](#) `Payload`
Command payload.

11.22.1 Detailed Description

Overwrite/Discard System Log Configuration Command Payload.

Definition at line 1146 of file `cfe_es_msg.h`.

11.22.2 Field Documentation

11.22.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_ES_OverWriteSysLogCmd::CommandHeader`

Command header.

Definition at line 1148 of file `cfe_es_msg.h`.

11.22.2.2 Payload

`CFE_ES_OverWriteSysLogCmd_Payload_t` `CFE_ES_OverWriteSysLogCmd::Payload`

Command payload.

Definition at line 1149 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.23 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `uint32 Mode`

`CFE_ES_LogMode_DISCARD`=Throw away most recent messages, `CFE_ES_LogMode_OVERWRITE`=Overwrite oldest with most recent

11.23.1 Detailed Description

Overwrite/Discard System Log Configuration Command Payload.

For command details, see [CFE_ES_OVER_WRITE_SYSLOG_CC](#)

Definition at line 1136 of file `cfe_es_msg.h`.

11.23.2 Field Documentation

11.23.2.1 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 1138 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.24 CFE_ES_PoolAlign Union Reference

Pool Alignment.

```
#include <cfes_api_typedefs.h>
```

Data Fields

- void * **Ptr**
Aligned pointer.
- long long int **LongInt**
Aligned Long Integer.
- long double **LongDouble**
Aligned Long Double.

11.24.1 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 93 of file cfe_es_api_typedefs.h.

11.24.2 Field Documentation

11.24.2.1 LongDouble

```
long double CFE_ES_PoolAlign::LongDouble
```

Aligned Long Double.

Definition at line 98 of file cfe_es_api_typedefs.h.

11.24.2.2 LongInt

```
long long int CFE_ES_PoolAlign::LongInt
```

Aligned Long Integer.

Definition at line 97 of file cfe_es_api_typedefs.h.

11.24.2.3 Ptr

```
void* CFE_ES_PoolAlign::Ptr
```

Aligned pointer.

Definition at line 95 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](#)

11.25 CFE_ES_PoolStatsTlm_Payload Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_ES_MemHandle_t PoolHandle](#)
Handle of memory pool whose stats are being telemetered.
- [CFE_ES_MemPoolStats_t PoolStats](#)
For more info, see [CFE_ES_MemPoolStats_t](#).

11.25.1 Detailed Description

Name Memory Pool Statistics Packet

Definition at line 1433 of file `cfe_es_msg.h`.

11.25.2 Field Documentation

11.25.2.1 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 1435 of file `cfe_es_msg.h`.

11.25.2.2 PoolStats

`CFE_ES_MemPoolStats_t` `CFE_ES_PoolStatsTlm_Payload::PoolStats`

For more info, see [CFE_ES_MemPoolStats_t](#).

Definition at line 1437 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.26 CFE_ES_ReloadAppCmd Struct Reference

Reload Application Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_ES_AppReloadCmd_Payload_t](#) `Payload`
Command payload.

11.26.1 Detailed Description

Reload Application Command.

Definition at line 1229 of file `cfe_es_msg.h`.

11.26.2 Field Documentation

11.26.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_ES_ReloadAppCmd::CommandHeader`

Command header.

Definition at line 1231 of file `cfe_es_msg.h`.

11.26.2.2 Payload

`CFE_ES_AppReloadCmd_Payload_t CFE_ES_ReloadAppCmd::Payload`

Command payload.

Definition at line 1232 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.27 CFE_ES_RestartCmd Struct Reference

Restart cFE Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_ES_RestartCmd_Payload_t Payload`
Command payload.

11.27.1 Detailed Description

Restart cFE Command.

Definition at line 1092 of file `cfe_es_msg.h`.

11.27.2 Field Documentation

11.27.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_ES_RestartCmd::CommandHeader`

Command header.

Definition at line 1094 of file `cfe_es_msg.h`.

11.27.2.2 Payload

```
CFE_ES_RestartCmd_Payload_t CFE_ES_RestartCmd::Payload
```

Command payload.

Definition at line 1095 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/cfe_es_msg.h

11.28 CFE_ES_RestartCmd_Payload Struct Reference

Restart cFE Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `uint16 RestartType`
`CFE_PSP_RST_TYPE_PROCESSOR`=Processor Reset or `CFE_PSP_RST_TYPE_POWERON`=Power-On Reset

11.28.1 Detailed Description

Restart cFE Command Payload.

For command details, see [CFE_ES_RESTART_CC](#)

Definition at line 1083 of file cfe_es_msg.h.

11.28.2 Field Documentation

11.28.2.1 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 1085 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/cfe_es_msg.h

11.29 CFE_ES_SendMemPoolStatsCmd Struct Reference

Send Memory Pool Statistics Command.

```
#include <cfes_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) **CommandHeader**
Command header.
- [CFE_ES_SendMemPoolStatsCmd_Payload_t](#) **Payload**
Command payload.

11.29.1 Detailed Description

Send Memory Pool Statistics Command.

Definition at line 1379 of file cfe_es_msg.h.

11.29.2 Field Documentation

11.29.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_SendMemPoolStatsCmd::CommandHeader

Command header.

Definition at line 1381 of file cfe_es_msg.h.

11.29.2.2 Payload

[CFE_ES_SendMemPoolStatsCmd_Payload_t](#) CFE_ES_SendMemPoolStatsCmd::Payload

Command payload.

Definition at line 1382 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfes_es_msg.h](#)

11.30 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference

Send Memory Pool Statistics Command Payload.

```
#include <cfes_es_msg.h>
```

Data Fields

- char [Application \[CFE_MISSION_MAX_API_LEN\]](#)
 - *RESERVED - should be all zeroes*
- [CFE_ES_MemHandle_t PoolHandle](#)
Handle of Pool whose statistics are to be telemetered.

11.30.1 Detailed Description

Send Memory Pool Statistics Command Payload.

For command details, see [CFE_ES_SEND_MEM_POOL_STATS_CC](#)

Definition at line 1369 of file cfe_es_msg.h.

11.30.2 Field Documentation

11.30.2.1 Application

```
char CFE_ES_SendMemPoolStatsCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]
```

- RESERVED - should be all zeroes

Definition at line 1371 of file cfe_es_msg.h.

11.30.2.2 PoolHandle

```
CFE_ES_MemHandle_t CFE_ES_SendMemPoolStatsCmd_Payload::PoolHandle
```

Handle of Pool whose statistics are to be telemetered.

Definition at line 1372 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfes_es_msg.h](#)

11.31 CFE_ES_SetMaxPRCountCmd Struct Reference

Set Maximum Processor Reset Count Command.

```
#include <cfes_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) **CommandHeader**
Command header.
- [CFE_ES_SetMaxPRCountCmd_Payload_t](#) **Payload**
Command payload.

11.31.1 Detailed Description

Set Maximum Processor Reset Count Command.

Definition at line 1250 of file cfe_es_msg.h.

11.31.2 Field Documentation

11.31.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) **CFE_ES_SetMaxPRCountCmd::CommandHeader**

Command header.

Definition at line 1252 of file cfe_es_msg.h.

11.31.2.2 Payload

[CFE_ES_SetMaxPRCountCmd_Payload_t](#) **CFE_ES_SetMaxPRCountCmd::Payload**

Command payload.

Definition at line 1253 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfes_es_msg.h](#)

11.32 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference

Set Maximum Processor Reset Count Command Payload.

```
#include <cfes_es_msg.h>
```

Data Fields

- `uint16 MaxPRCount`

New maximum number of Processor Resets before an automatic Power-On Reset is performed.

11.32.1 Detailed Description

Set Maximum Processor Reset Count Command Payload.

For command details, see [CFE_ES_SET_MAX_PR_COUNT_CC](#)

Definition at line 1241 of file cfe_es_msg.h.

11.32.2 Field Documentation

11.32.2.1 MaxPRCount

```
uint16 CFE_ES_SetMaxPRCountCmd_Payload::MaxPRCount
```

New maximum number of Processor Resets before an automatic Power-On Reset is performed.

Definition at line 1243 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfes_es_msg.h](#)

11.33 CFE_ES_SetPerfFilterMaskCmd Struct Reference

Set Performance Analyzer Filter Mask Command.

```
#include <cfes_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_SetPerfFilterMaskCmd_Payload_t](#) Payload
Command payload.

11.33.1 Detailed Description

Set Performance Analyzer Filter Mask Command.

Definition at line 1335 of file cfe_es_msg.h.

11.33.2 Field Documentation

11.33.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_SetPerfFilterMaskCmd::CommandHeader

Command header.

Definition at line 1337 of file cfe_es_msg.h.

11.33.2.2 Payload

[CFE_ES_SetPerfFilterMaskCmd_Payload_t](#) CFE_ES_SetPerfFilterMaskCmd::Payload

Command payload.

Definition at line 1338 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.34 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference

Set Performance Analyzer Filter Mask Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `uint32 FilterMaskNum`
Index into array of Filter Masks.
- `uint32 FilterMask`
New Mask for specified entry in array of Filter Masks.

11.34.1 Detailed Description

Set Performance Analyzer Filter Mask Command Payload.

For command details, see [CFE_ES_SET_PERF_FILTER_MASK_CC](#)

Definition at line 1325 of file cfe_es_msg.h.

11.34.2 Field Documentation

11.34.2.1 FilterMask

`uint32 CFE_ES_SetPerfFilterMaskCmd_Payload::FilterMask`

New Mask for specified entry in array of Filter Masks.

Definition at line 1328 of file cfe_es_msg.h.

11.34.2.2 FilterMaskNum

`uint32 CFE_ES_SetPerfFilterMaskCmd_Payload::FilterMaskNum`

Index into array of Filter Masks.

Definition at line 1327 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.35 CFE_ES_SetPerfTriggerMaskCmd Struct Reference

Set Performance Analyzer Trigger Mask Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_SetPerfTrigMaskCmd_Payload_t](#) Payload
Command payload.

11.35.1 Detailed Description

Set Performance Analyzer Trigger Mask Command.

Definition at line 1357 of file cfe_es_msg.h.

11.35.2 Field Documentation

11.35.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_SetPerfTriggerMaskCmd::CommandHeader

Command header.

Definition at line 1359 of file cfe_es_msg.h.

11.35.2.2 Payload

[CFE_ES_SetPerfTrigMaskCmd_Payload_t](#) CFE_ES_SetPerfTriggerMaskCmd::Payload

Command payload.

Definition at line 1360 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.36 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference

Set Performance Analyzer Trigger Mask Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `uint32 TriggerMaskNum`
Index into array of Trigger Masks.
- `uint32 TriggerMask`
New Mask for specified entry in array of Trigger Masks.

11.36.1 Detailed Description

Set Performance Analyzer Trigger Mask Command Payload.

For command details, see [CFE_ES_SET_PERF_TRIGGER_MASK_CC](#)

Definition at line 1347 of file cfe_es_msg.h.

11.36.2 Field Documentation

11.36.2.1 TriggerMask

`uint32 CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMask`

New Mask for specified entry in array of Trigger Masks.

Definition at line 1350 of file cfe_es_msg.h.

11.36.2.2 TriggerMaskNum

`uint32 CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMaskNum`

Index into array of Trigger Masks.

Definition at line 1349 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.37 CFE_ES_StartApp Struct Reference

Start Application Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_StartAppCmd_Payload_t](#) Payload
Command payload.

11.37.1 Detailed Description

Start Application Command.

Definition at line 1178 of file cfe_es_msg.h.

11.37.2 Field Documentation

11.37.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_ES_StartApp::CommandHeader

Command header.

Definition at line 1180 of file cfe_es_msg.h.

11.37.2.2 Payload

[CFE_ES_StartAppCmd_Payload_t](#) CFE_ES_StartApp::Payload

Command payload.

Definition at line 1181 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.38 CFE_ES_StartAppCmd_Payload Struct Reference

Start Application Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char [Application](#) [[CFE_MISSION_MAX_API_LEN](#)]
Name of Application to be started.
- char [AppEntryPoint](#) [[CFE_MISSION_MAX_API_LEN](#)]
Symbolic name of Application's entry point.
- char [AppFileName](#) [[CFE_MISSION_MAX_PATH_LEN](#)]
Full path and filename of Application's executable image.
- [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_RESET=On exception, perform a Processor Reset
- [CFE_ES_TaskPriority_Atom_t](#) [Priority](#)
The new Applications runtime priority.

11.38.1 Detailed Description

Start Application Command Payload.

For command details, see [CFE_ES_START_APP_CC](#)

Definition at line 1158 of file cfe_es_msg.h.

11.38.2 Field Documentation

11.38.2.1 AppEntryPoint

```
char CFE_ES_StartAppCmd_Payload::AppEntryPoint[CFE\_MISSION\_MAX\_API\_LEN]
```

Symbolic name of Application's entry point.

Definition at line 1161 of file cfe_es_msg.h.

11.38.2.2 AppFileName

```
char CFE_ES_StartAppCmd_Payload::AppFileName[CFE\_MISSION\_MAX\_PATH\_LEN]
```

Full path and filename of Application's executable image.

Definition at line 1162 of file cfe_es_msg.h.

11.38.2.3 Application

```
char CFE_ES_StartAppCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]
```

Name of Application to be started.

Definition at line 1160 of file cfe_es_msg.h.

11.38.2.4 ExceptionAction

```
CFE_ES_ExceptionAction_Enum_t CFE_ES_StartAppCmd_Payload::ExceptionAction
```

CFE_ES_ExceptionAction_RESTART_APP=On exception, restart Application, **CFE_ES_ExceptionAction_PROC_RESET**=On exception, perform a Processor Reset

Definition at line 1167 of file cfe_es_msg.h.

11.38.2.5 Priority

```
CFE_ES_TaskPriority_Atom_t CFE_ES_StartAppCmd_Payload::Priority
```

The new Applications runtime priority.

Definition at line 1171 of file cfe_es_msg.h.

11.38.2.6 StackSize

```
CFE_ES_MemOffset_t CFE_ES_StartAppCmd_Payload::StackSize
```

Desired stack size for the new application.

Definition at line 1165 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/cfe_es_msg.h

11.39 CFE_ES_StartPerfCmd_Payload Struct Reference

Start Performance Analyzer Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- `uint32 TriggerMode`

Desired trigger position (Start, Center, End)

11.39.1 Detailed Description

Start Performance Analyzer Command Payload.

For command details, see [CFE_ES_START_PERF_DATA_CC](#)

Definition at line 1284 of file cfe_es_msg.h.

11.39.2 Field Documentation

11.39.2.1 TriggerMode

`uint32 CFE_ES_StartPerfCmd_Payload::TriggerMode`

Desired trigger position (Start, Center, End)

Definition at line 1286 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.40 CFE_ES_StartPerfDataCmd Struct Reference

Start Performance Analyzer Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

Command header.

- `CFE_ES_StartPerfCmd_Payload_t Payload`

Command payload.

11.40.1 Detailed Description

Start Performance Analyzer Command.

Definition at line 1292 of file `cfe_es_msg.h`.

11.40.2 Field Documentation

11.40.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_ES_StartPerfDataCmd::CommandHeader`

Command header.

Definition at line 1294 of file `cfe_es_msg.h`.

11.40.2.2 Payload

`CFE_ES_StartPerfCmd_Payload_t CFE_ES_StartPerfDataCmd::Payload`

Command payload.

Definition at line 1295 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.41 CFE_ES_StopPerfCmd_Payload Struct Reference

Stop Performance Analyzer Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char `DataFileName [CFE_MISSION_MAX_PATH_LEN]`

ASCII text string of full path and filename of file Performance Analyzer data is to be written.

11.41.1 Detailed Description

Stop Performance Analyzer Command Payload.

For command details, see [CFE_ES_STOP_PERF_DATA_CC](#)

Definition at line 1304 of file cfe_es_msg.h.

11.41.2 Field Documentation

11.41.2.1 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 1306 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/fsw/inc/[cfe_es_msg.h](#)

11.42 CFE_ES_StopPerfDataCmd Struct Reference

Stop Performance Analyzer Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_ES_StopPerfCmd_Payload_t](#) Payload
Command payload.

11.42.1 Detailed Description

Stop Performance Analyzer Command.

Definition at line 1313 of file cfe_es_msg.h.

11.42.2 Field Documentation

11.42.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_ES_StopPerfDataCmd::CommandHeader`

Command header.

Definition at line 1315 of file `cfe_es_msg.h`.

11.42.2.2 Payload

`CFE_ES_StopPerfCmd_Payload_t` `CFE_ES_StopPerfDataCmd::Payload`

Command payload.

Definition at line 1316 of file `cfe_es_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/fsw/inc/cfe_es_msg.h`

11.43 CFE_ES_TaskInfo Struct Reference

Task Information.

```
#include <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 Appld`
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]`

11.43.1 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 501 of file `cfe_es_extern_typedefs.h`.

11.43.2 Field Documentation

11.43.2.1 AppId

`CFE_ES_AppId_t` `CFE_ES_TaskInfo::AppId`

Parent Application ID.

Definition at line 506 of file `cfe_es_extern_typedefs.h`.

11.43.2.2 AppName

`char` `CFE_ES_TaskInfo::AppName[CFE_MISSION_MAX_API_LEN]`

Parent Application Name.

Definition at line 507 of file `cfe_es_extern_typedefs.h`.

11.43.2.3 ExecutionCounter

`uint32` `CFE_ES_TaskInfo::ExecutionCounter`

Task Execution Counter.

Definition at line 504 of file `cfe_es_extern_typedefs.h`.

Referenced by `HS_HousekeepingReq()`.

11.43.2.4 Priority

`CFE_ES_TaskPriority_Atom_t CFE_ES_TaskInfo::Priority`

Priority of task

Definition at line 509 of file `cfe_es_extern_typedefs.h`.

11.43.2.5 Spare

`uint8 CFE_ES_TaskInfo::Spare[2]`

Spare bytes for alignment

Definition at line 510 of file `cfe_es_extern_typedefs.h`.

11.43.2.6 StackSize

`CFE_ES_MemOffset_t CFE_ES_TaskInfo::StackSize`

Size of task stack

Definition at line 508 of file `cfe_es_extern_typedefs.h`.

11.43.2.7 TaskId

`CFE_ES_TaskId_t CFE_ES_TaskInfo::TaskId`

Task Id.

Definition at line 503 of file `cfe_es_extern_typedefs.h`.

11.43.2.8 TaskName

`char CFE_ES_TaskInfo::TaskName[CFE_MISSION_MAX_API_LEN]`

Task Name.

Definition at line 505 of file `cfe_es_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h`

11.44 CFE_EVS_AppDataCmd_Payload Struct Reference

Write Event Services Application Information to File Command Payload.

```
#include <cfе_evs_msg.h>
```

Data Fields

- char AppDataFilename [CFE_MISSION_MAX_PATH_LEN]

Filename where application data is to be written.

11.44.1 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 945 of file cfe_evs_msg.h.

11.44.2 Field Documentation

11.44.2.1 AppDataFilename

```
char CFE_EVS_AppDataCmd_Payload::AppDataFilename [CFE_MISSION_MAX_PATH_LEN]
```

Filename where application data is to be written.

Definition at line 947 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.45 CFE_EVS_AppNameBitMaskCmd Struct Reference

Generic App Name and Bitmask Command.

```
#include <cfе_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_AppNameBitMaskCmd_Payload_t](#) Payload
Command payload.

11.45.1 Detailed Description

Generic App Name and Bitmask Command.

Definition at line 1109 of file cfe_evs_msg.h.

11.45.2 Field Documentation

11.45.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_EVS_AppNameBitMaskCmd::CommandHeader

Command header.

Definition at line 1111 of file cfe_evs_msg.h.

11.45.2.2 Payload

[CFE_EVS_AppNameBitMaskCmd_Payload_t](#) CFE_EVS_AppNameBitMaskCmd::Payload

Command payload.

Definition at line 1112 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.46 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference

Generic App Name and Bitmask Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char `AppName [CFE_MISSION_MAX_API_LEN]`
Application name to use in the command.
- `uint8 BitMask`
BitMask to use in the command.
- `uint8 Spare`
Pad to even byte.

11.46.1 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 1099 of file cfe_evs_msg.h.

11.46.2 Field Documentation

11.46.2.1 AppName

```
char CFE_EVS_AppNameBitMaskCmd_Payload::AppName[CFE_MISSION_MAX_API_LEN]
```

Application name to use in the command.

Definition at line 1101 of file cfe_evs_msg.h.

11.46.2.2 BitMask

```
uint8 CFE_EVS_AppNameBitMaskCmd_Payload::BitMask
```

BitMask to use in the command.

Definition at line 1102 of file cfe_evs_msg.h.

11.46.2.3 Spare

```
uint8 CFE_EVS_AppNameBitMaskCmd_Payload::Spare
```

Pad to even byte.

Definition at line 1103 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/evs/fsw/inc/cfe_evs_msg.h](#)

11.47 CFE_EVS_AppNameCmd Struct Reference

Generic App Name Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_EVS_AppNameCmd_Payload_t Payload](#)
Command payload.

11.47.1 Detailed Description

Generic App Name Command.

Definition at line 1048 of file cfe_evs_msg.h.

11.47.2 Field Documentation

11.47.2.1 CommandHeader

```
CFE_MSG_CommandHeader_t CFE_EVS_AppNameCmd::CommandHeader
```

Command header.

Definition at line 1050 of file cfe_evs_msg.h.

11.47.2.2 Payload

`CFE_EVS_AppNameCmd_Payload_t` `CFE_EVS_AppNameCmd::Payload`

Command payload.

Definition at line 1051 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.48 CFE_EVS_AppNameCmd_Payload Struct Reference

Generic App Name Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- `char AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

11.48.1 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 1040 of file `cfe_evs_msg.h`.

11.48.2 Field Documentation

11.48.2.1 AppName

```
char CFE_EVS_AppNameCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]
```

Application name to use in the command.

Definition at line 1042 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.49 CFE_EVS_AppNameEventIDCmd Struct Reference

Generic App Name and Event ID Command.

```
#include <cfе_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_EVS_AppNameEventIDCmd_Payload_t](#) `Payload`
Command payload.

11.49.1 Detailed Description

Generic App Name and Event ID Command.

Definition at line 1079 of file `cfе_evs_msg.h`.

11.49.2 Field Documentation

11.49.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) `CFE_EVS_AppNameEventIDCmd::CommandHeader`

Command header.

Definition at line 1081 of file `cfе_evs_msg.h`.

11.49.2.2 Payload

[CFE_EVS_AppNameEventIDCmd_Payload_t](#) `CFE_EVS_AppNameEventIDCmd::Payload`

Command payload.

Definition at line 1082 of file `cfе_evs_msg.h`.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfе_evs_msg.h](#)

11.50 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference

Generic App Name and Event ID Command Payload.

```
#include <cfе_evs_msg.h>
```

Data Fields

- `charAppName [CFE_MISSION_MAX_API_LEN]`
Application name to use in the command.
- `uint16EventID`
Event ID to use in the command.

11.50.1 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 1070 of file cfe_evs_msg.h.

11.50.2 Field Documentation

11.50.2.1 AppName

```
char CFE_EVS_AppNameEventIDCmd_Payload::AppName[CFE_MISSION_MAX_API_LEN]
```

Application name to use in the command.

Definition at line 1072 of file cfe_evs_msg.h.

11.50.2.2 EventID

```
uint16 CFE_EVS_AppNameEventIDCmd_Payload::EventID
```

Event ID to use in the command.

Definition at line 1073 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfе_evs_msg.h](#)

11.51 CFE_EVS_AppNameEventIDMaskCmd Struct Reference

Generic App Name, Event ID, Mask Command.

```
#include <cfе_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_EVS_AppNameEventIDMaskCmd_Payload_t](#) `Payload`
Command payload.

11.51.1 Detailed Description

Generic App Name, Event ID, Mask Command.

Definition at line 1140 of file `cfе_evs_msg.h`.

11.51.2 Field Documentation

11.51.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) `CFE_EVS_AppNameEventIDMaskCmd::CommandHeader`

Command header.

Definition at line 1142 of file `cfе_evs_msg.h`.

11.51.2.2 Payload

[CFE_EVS_AppNameEventIDMaskCmd_Payload_t](#) `CFE_EVS_AppNameEventIDMaskCmd::Payload`

Command payload.

Definition at line 1143 of file `cfе_evs_msg.h`.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfе_evs_msg.h](#)

11.52 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference

Generic App Name, Event ID, Mask Command Payload.

```
#include <cfе_evs_msg.h>
```

Data Fields

- char [AppName \[CFE_MISSION_MAX_API_LEN\]](#)
Application name to use in the command.
- [uint16 EventID](#)
Event ID to use in the command.
- [uint16 Mask](#)
Mask to use in the command.

11.52.1 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 1130 of file cfe_evs_msg.h.

11.52.2 Field Documentation

11.52.2.1 AppName

```
char CFE_EVS_AppNameEventIDMaskCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]
```

Application name to use in the command.

Definition at line 1132 of file cfe_evs_msg.h.

11.52.2.2 EventID

```
uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::EventID
```

Event ID to use in the command.

Definition at line 1133 of file cfe_evs_msg.h.

11.52.2.3 Mask

```
uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::Mask
```

Mask to use in the command.

Definition at line 1134 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.53 CFE_EVS_AppTlmData Struct Reference

```
#include <cfe_evs_msg.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 Padding](#)
Padding for 32 bit boundary.

11.53.1 Detailed Description

Definition at line 1158 of file cfe_evs_msg.h.

11.53.2 Field Documentation

11.53.2.1 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 1164 of file cfe_evs_msg.h.

11.53.2.2 ApplID

`CFE_ES_AppId_t` `CFE_EVS_AppTlmData::ApplID`

Numerical application identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPID

Definition at line 1160 of file `cfe_evs_msg.h`.

11.53.2.3 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 1162 of file `cfe_evs_msg.h`.

11.53.2.4 Padding

`uint8` `CFE_EVS_AppTlmData::Padding`

Padding for 32 bit boundary.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].SPARE2ALIGN3

Definition at line 1166 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.54 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.

11.54.1 Detailed Description

Event message filter definition structure.

Definition at line 60 of file `cfe_evs_api_typedefs.h`.

11.54.2 Field Documentation

11.54.2.1 EventID

`uint16 CFE_EVS_BinFilter::EventID`

Numerical event identifier.

Definition at line 62 of file `cfe_evs_api_typedefs.h`.

11.54.2.2 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`

11.55 CFE_EVS_BitMaskCmd Struct Reference

Generic Bitmask Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_EVS_BitMaskCmd_Payload_t](#) Payload
Command payload.

11.55.1 Detailed Description

Generic Bitmask Command.

Definition at line 1017 of file cfe_evs_msg.h.

11.55.2 Field Documentation

11.55.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_EVS_BitMaskCmd::CommandHeader

Command header.

Definition at line 1019 of file cfe_evs_msg.h.

11.55.2.2 Payload

[CFE_EVS_BitMaskCmd_Payload_t](#) CFE_EVS_BitMaskCmd::Payload

Command payload.

Definition at line 1020 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.56 CFE_EVS_BitMaskCmd_Payload Struct Reference

Generic Bitmask Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- `uint8 BitMask`
BitMask to use in the command.
- `uint8 Spare`
Pad to even byte.

11.56.1 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 1008 of file `cfe_evs_msg.h`.

11.56.2 Field Documentation

11.56.2.1 BitMask

`uint8 CFE_EVS_BitMaskCmd_Payload::BitMask`

BitMask to use in the command.

Definition at line 1010 of file `cfe_evs_msg.h`.

11.56.2.2 Spare

`uint8 CFE_EVS_BitMaskCmd_Payload::Spare`

Pad to even byte.

Definition at line 1011 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- [cfe/modules/evs/fsw/inc/cfe_evs_msg.h](#)

11.57 CFE_EVS_HousekeepingTim Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_EVS_HousekeepingTlm_Payload_t](#) Payload
Telemetry payload.

11.57.1 Detailed Description

Definition at line 1213 of file cfe_evs_msg.h.

11.57.2 Field Documentation

11.57.2.1 Payload

[CFE_EVS_HousekeepingTlm_Payload_t](#) CFE_EVS_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 1216 of file cfe_evs_msg.h.

11.57.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_EVS_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 1215 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.58 CFE_EVS_HousekeepingTlm_Payload Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- **uint8 CommandCounter**
EVS Command Counter.
- **uint8 CommandErrorCounter**
EVS Command Error Counter.
- **uint8 MessageFormatMode**
Event message format mode (short/long)
- **uint8 MessageTruncCounter**
Event message truncation counter.
- **uint8 UnregisteredAppCounter**
Unregistered application message send counter.
- **uint8 OutputPort**
Output port mask.
- **uint8 LogFullFlag**
Local event log full flag.
- **uint8 LogMode**
Local event logging mode (overwrite/discard)
- **uint16 MessageSendCounter**
Event message send counter.
- **uint16 LogOverflowCounter**
Local event log overflow counter.
- **uint8 LogEnabled**
Current event log enable/disable state.
- **uint8 Spare1**
Padding for 32 bit boundary.
- **uint8 Spare2**
Padding for 32 bit boundary.
- **uint8 Spare3**
Padding for 32 bit boundary.
- **CFE_EVS_AppTlmData_t AppData [CFE_MISSION_ES_MAX_APPLICATIONS]**
Array of registered application table data.

11.58.1 Detailed Description

Name Event Services Housekeeping Telemetry Packet

Definition at line 1174 of file cfe_evs_msg.h.

11.58.2 Field Documentation

11.58.2.1 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 1208 of file cfe_evs_msg.h.

11.58.2.2 CommandCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload::CommandCounter
```

EVS Command Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_CMDPC

Definition at line 1176 of file cfe_evs_msg.h.

11.58.2.3 CommandErrorCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload::CommandErrorCounter
```

EVS Command Error Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_CMDEC

Definition at line 1178 of file cfe_evs_msg.h.

11.58.2.4 LogEnabled

```
uint8 CFE_EVS_HousekeepingTlm_Payload::LogEnabled
```

Current event log enable/disable state.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_LOGENABLED

Definition at line 1199 of file cfe_evs_msg.h.

11.58.2.5 LogFullFlag

```
uint8 CFE_EVS_HousekeepingTlm_Payload::LogFullFlag
```

Local event log full flag.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_LOGFULL

Definition at line 1189 of file cfe_evs_msg.h.

11.58.2.6 LogMode

```
uint8 CFE_EVS_HousekeepingTlm_Payload::LogMode
```

Local event logging mode (overwrite/discard)

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_LOGMODE

Definition at line 1191 of file cfe_evs_msg.h.

11.58.2.7 LogOverflowCounter

```
uint16 CFE_EVS_HousekeepingTlm_Payload::LogOverflowCounter
```

Local event log overflow counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_LOGOVERFLOWC

Definition at line 1196 of file cfe_evs_msg.h.

11.58.2.8 MessageFormatMode

```
uint8 CFE_EVS_HousekeepingTlm_Payload::MessageFormatMode
```

Event message format mode (short/long)

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_MSGFMTMODE

Definition at line 1180 of file cfe_evs_msg.h.

11.58.2.9 MessageSendCounter

```
uint16 CFE_EVS_HousekeepingTlm_Payload::MessageSendCounter
```

Event message send counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_MSGSENTC

Definition at line 1194 of file cfe_evs_msg.h.

11.58.2.10 MessageTruncCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload::MessageTruncCounter
```

Event message truncation counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_MSGTRUNC

Definition at line 1182 of file cfe_evs_msg.h.

11.58.2.11 OutputPort

```
uint8 CFE_EVS_HousekeepingTlm_Payload::OutputPort
```

Output port mask.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_OUTPUTPORT

Definition at line 1187 of file cfe_evs_msg.h.

11.58.2.12 Spare1

```
uint8 CFE_EVS_HousekeepingTlm_Payload::Spare1
```

Padding for 32 bit boundary.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_HK_SPARE1

Definition at line 1201 of file cfe_evs_msg.h.

11.58.2.13 Spare2

```
uint8 CFE_EVS_HousekeepingTlm_Payload::Spare2
```

Padding for 32 bit boundary.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_HK_SPARE2

Definition at line 1203 of file cfe_evs_msg.h.

11.58.2.14 Spare3

```
uint8 CFE_EVS_HousekeepingTlm_Payload::Spare3
```

Padding for 32 bit boundary.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_HK_SPARE3

Definition at line 1205 of file cfe_evs_msg.h.

11.58.2.15 UnregisteredAppCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload::UnregisteredAppCounter
```

Unregistered application message send counter.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_UNREGAPPC

Definition at line 1185 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.59 CFE_EVS_LogFileCmd_Payload Struct Reference

Write Event Log to File Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char [LogFilename \[CFE_MISSION_MAX_PATH_LEN\]](#)

Filename where log data is to be written.

11.59.1 Detailed Description

Write Event Log to File Command Payload.

For command details, see [CFE_EVS_WRITE_LOG_DATA_FILE_CC](#)

Definition at line 925 of file cfe_evs_msg.h.

11.59.2 Field Documentation**11.59.2.1 LogFilename**

```
char CFE_EVS_LogFileCmd_Payload::LogFilename[CFE_MISSION_MAX_PATH_LEN]
```

Filename where log data is to be written.

Definition at line 927 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.60 CFE_EVS_LongEventTlm Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_EVS_LongEventTlm_Payload_t Payload](#)
Telemetry payload.

11.60.1 Detailed Description

Definition at line 1259 of file cfe_evs_msg.h.

11.60.2 Field Documentation

11.60.2.1 Payload

[CFE_EVS_LongEventTlm_Payload_t](#) CFE_EVS_LongEventTlm::Payload

Telemetry payload.

Definition at line 1262 of file cfe_evs_msg.h.

Referenced by HS_MonitorEvent().

11.60.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_EVS_LongEventTlm::TelemetryHeader

Telemetry header.

Definition at line 1261 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.61 CFE_EVS_LongEventTlm_Payload Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_EVS_PacketID_t](#) PacketID
Event packet information.
- char Message [[CFE_MISSION_EVS_MAX_MESSAGE_LENGTH](#)]
Event message string.
- uint8 Spare1
Structure padding.
- uint8 Spare2
Structure padding.

11.61.1 Detailed Description

Name Event Message Telemetry Packet (Long format)

Definition at line 1239 of file cfe_evs_msg.h.

11.61.2 Field Documentation

11.61.2.1 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 1242 of file cfe_evs_msg.h.

11.61.2.2 PacketID

`CFE_EVS_PacketID_t CFE_EVS_LongEventTlm_Payload::PacketID`

Event packet information.

Definition at line 1241 of file cfe_evs_msg.h.

Referenced by HS_MonitorEvent().

11.61.2.3 Spare1

`uint8 CFE_EVS_LongEventTlm_Payload::Spare1`

Structure padding.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_SPARE1

Definition at line 1244 of file cfe_evs_msg.h.

11.61.2.4 Spare2

`uint8 CFE_EVS_LongEventTlm_Payload::Spare2`

Structure padding.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_SPARE2

Definition at line 1246 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.62 CFE_EVS_NoArgsCmd Struct Reference

Command with no additional arguments.

```
#include <cf_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.

11.62.1 Detailed Description

Command with no additional arguments.

Definition at line 905 of file cfe_evs_msg.h.

11.62.2 Field Documentation

11.62.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_EVS_NoArgsCmd::CommandHeader

Command header.

Definition at line 907 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.63 CFE_EVS_PacketID Struct Reference

```
#include <cf_evs_msg.h>
```

Data Fields

- char [AppName \[CFE_MISSION_MAX_API_LEN\]](#)
Application name.
- [uint16 EventID](#)
Numerical event identifier.
- [uint16 EventType](#)
Numerical event type identifier.
- [uint32 SpacecraftID](#)
Spacecraft identifier.
- [uint32 ProcessorID](#)
Numerical processor identifier.

11.63.1 Detailed Description

Telemetry packet structures

Definition at line 1221 of file cfe_evs_msg.h.

11.63.2 Field Documentation

11.63.2.1 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 1223 of file cfe_evs_msg.h.

Referenced by HS_MonitorEvent().

11.63.2.2 EventID

```
uint16 CFE_EVS_PacketID::EventID
```

Numerical event identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_EVENTID

Definition at line 1225 of file cfe_evs_msg.h.

Referenced by HS_MonitorEvent().

11.63.2.3 EventType

```
uint16 CFE_EVS_PacketID::EventType
```

Numerical event type identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_EVENTTYPE

Definition at line 1227 of file cfe_evs_msg.h.

11.63.2.4 ProcessorID

`uint32 CFE_EVS_PacketID::ProcessorID`

Numerical processor identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_PROCESSORID

Definition at line 1231 of file cfe_evs_msg.h.

11.63.2.5 SpacecraftID

`uint32 CFE_EVS_PacketID::SpacecraftID`

Spacecraft identifier.

Telemetry Mnemonic(s) \$sc_\$cpu_EVS_SCID

Definition at line 1229 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.64 CFE_EVS_SetEventFormatCode_Payload Struct Reference

Set Event Format Mode Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- **CFE_EVS_MsgFormat_Enum_t MsgFormat**
Mode to use in the command.
- **uint8 Spare**
Pad to even byte.

11.64.1 Detailed Description

Set Event Format Mode Command Payload.

For command details, see [CFE_EVS_SET_EVENT_FORMAT_MODE_CC](#)

Definition at line 986 of file cfe_evs_msg.h.

11.64.2 Field Documentation

11.64.2.1 MsgFormat

```
CFE_EVS_MsgFormat_Enum_t CFE_EVS_SetEventFormatCode_Payload::MsgFormat
```

Mode to use in the command.

Definition at line 988 of file cfe_evs_msg.h.

11.64.2.2 Spare

```
uint8 CFE_EVS_SetEventFormatCode_Payload::Spare
```

Pad to even byte.

Definition at line 989 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.65 CFE_EVS_SetEventFormatModeCmd Struct Reference

Set Event Format Mode Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_EVS_SetEventFormatMode_Payload_t Payload](#)
Command payload.

11.65.1 Detailed Description

Set Event Format Mode Command.

Definition at line 995 of file cfe_evs_msg.h.

11.65.2 Field Documentation

11.65.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_EVS_SetEventFormatModeCmd::CommandHeader

Command header.

Definition at line 997 of file cfe_evs_msg.h.

11.65.2.2 Payload

[CFE_EVS_SetEventFormatMode_Payload_t](#) CFE_EVS_SetEventFormatModeCmd::Payload

Command payload.

Definition at line 998 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.66 CFE_EVS_SetLogMode_Payload Struct Reference

Set Log Mode Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_EVS_LogMode_Enum_t LogMode](#)
Mode to use in the command.
- [uint8 Spare](#)
Pad to even byte.

11.66.1 Detailed Description

Set Log Mode Command Payload.

For command details, see [CFE_EVS_SET_LOG_MODE_CC](#)

Definition at line 965 of file cfe_evs_msg.h.

11.66.2 Field Documentation

11.66.2.1 LogMode

`CFE_EVS_LogMode_Enum_t` `CFE_EVS_SetLogMode_Payload::LogMode`

Mode to use in the command.

Definition at line 967 of file `cfe_evs_msg.h`.

11.66.2.2 Spare

`uint8` `CFE_EVS_SetLogMode_Payload::Spare`

Pad to even byte.

Definition at line 968 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.67 CFE_EVS_SetLogModeCmd Struct Reference

Set Log Mode Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t` `CommandHeader`
Command header.
- `CFE_EVS_SetLogMode_Payload_t` `Payload`
Command payload.

11.67.1 Detailed Description

Set Log Mode Command.

Definition at line 974 of file `cfe_evs_msg.h`.

11.67.2 Field Documentation

11.67.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_EVS_SetLogModeCmd::CommandHeader

Command header.

Definition at line 976 of file cfe_evs_msg.h.

11.67.2.2 Payload

[CFE_EVS_SetLogMode_Payload_t](#) CFE_EVS_SetLogModeCmd::Payload

Command payload.

Definition at line 977 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/[cfe_evs_msg.h](#)

11.68 CFE_EVS_ShortEventTlm Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t](#) TelemetryHeader
Telemetry header.
- [CFE_EVS_ShortEventTlm_Payload_t](#) Payload
Telemetry payload.

11.68.1 Detailed Description

Definition at line 1266 of file cfe_evs_msg.h.

11.68.2 Field Documentation

11.68.2.1 Payload

`CFE_EVS_ShortEventTlm_Payload_t` `CFE_EVS_ShortEventTlm::Payload`

Telemetry payload.

Definition at line 1269 of file `cfe_evs_msg.h`.

11.68.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` `CFE_EVS_ShortEventTlm::TelemetryHeader`

Telemetry header.

Definition at line 1268 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.69 CFE_EVS_ShortEventTlm_Payload Struct Reference

#include <`cfe_evs_msg.h`>

Data Fields

- `CFE_EVS_PacketID_t PacketID`

Event packet information.

11.69.1 Detailed Description

Name Event Message Telemetry Packet (Short format)

Definition at line 1253 of file `cfe_evs_msg.h`.

11.69.2 Field Documentation

11.69.2.1 PacketID

`CFE_EVS_PacketID_t CFE_EVS_ShortEventTlm_Payload::PacketID`

Event packet information.

Definition at line 1255 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.70 CFE_EVS_WriteAppDataFileCmd Struct Reference

Write Event Services Application Information to File Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_AppDataCmd_Payload_t Payload`
Command payload.

11.70.1 Detailed Description

Write Event Services Application Information to File Command.

Definition at line 953 of file `cfe_evs_msg.h`.

11.70.2 Field Documentation

11.70.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_EVS_WriteAppDataFileCmd::CommandHeader`

Command header.

Definition at line 955 of file `cfe_evs_msg.h`.

11.70.2.2 Payload

`CFE_EVS_AppDataCmd_Payload_t CFE_EVS_WriteAppDataFileCmd::Payload`

Command payload.

Definition at line 956 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.71 CFE_EVS_WriteLogFileCmd Struct Reference

Write Event Log to File Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_EVS_LogFileCmd_Payload_t Payload`
Command payload.

11.71.1 Detailed Description

Write Event Log to File Command.

Definition at line 933 of file `cfe_evs_msg.h`.

11.71.2 Field Documentation

11.71.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_EVS_WriteLogFileCmd::CommandHeader`

Command header.

Definition at line 935 of file `cfe_evs_msg.h`.

11.71.2.2 Payload

`CFE_EVS_LogFileCmd_Payload_t` `CFE_EVS_WriteLogFileCmd::Payload`

Command payload.

Definition at line 936 of file `cfe_evs_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/fsw/inc/cfe_evs_msg.h`

11.72 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` [`OS_MAX_PATH_LEN`]
- `uint32 FileSubType`
- char `Description` [`CFE_FS_HDR_DESC_MAX_LEN`]
- `CFE_FS_FileWriteGetData_t GetData`
- `CFE_FS_FileWriteOnEvent_t OnEvent`

11.72.1 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`.

11.72.2 Field Documentation

11.72.2.1 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`.

11.72.2.2 FileName

```
char CFE_FS_FileWriteMetaData::FileName[OS_MAX_PATH_LEN]
```

Name of file to write

Definition at line 128 of file cfe_fs_api_typedefs.h.

11.72.2.3 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.

11.72.2.4 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.

11.72.2.5 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.

11.72.2.6 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

11.73 CFE_FS_Header Struct Reference

Standard cFE File header structure definition.

```
#include <cfef_s_extern_typedefs.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.

11.73.1 Detailed Description

Standard cFE File header structure definition.

Definition at line 204 of file cfe_fs_extern_typedefs.h.

11.73.2 Field Documentation

11.73.2.1 ApplicationID

```
uint32 CFE_FS_Header::ApplicationID
```

Application that generated the file.

Definition at line 213 of file cfe_fs_extern_typedefs.h.

11.73.2.2 ContentType

```
uint32 CFE_FS_Header::ContentType
```

Identifies the content type ('cFE1'=0x63464531)

Definition at line 206 of file cfe_fs_extern_typedefs.h.

11.73.2.3 Description

```
char CFE_FS_Header::Description[CFE_FS_HDR_DESC_MAX_LEN]
```

File description.

Definition at line 218 of file cfe_fs_extern_typedefs.h.

11.73.2.4 Length

```
uint32 CFE_FS_Header::Length
```

Length of this header to support external processing.

Definition at line 210 of file cfe_fs_extern_typedefs.h.

11.73.2.5 ProcessorID

```
uint32 CFE_FS_Header::ProcessorID
```

Processor that generated the file.

Definition at line 212 of file cfe_fs_extern_typedefs.h.

11.73.2.6 SpacecraftID

```
uint32 CFE_FS_Header::SpacecraftID
```

Spacecraft that generated the file.

Definition at line 211 of file cfe_fs_extern_typedefs.h.

11.73.2.7 SubType

`uint32 CFE_FS_Header::SubType`

Type of ContentType, if necessary.

Standard SubType definitions can be found [here](#)

Definition at line 207 of file `cfe_fs_extern_typedefs.h`.

11.73.2.8 TimeSeconds

`uint32 CFE_FS_Header::TimeSeconds`

File creation timestamp (seconds)

Definition at line 215 of file `cfe_fs_extern_typedefs.h`.

11.73.2.9 TimeSubSeconds

`uint32 CFE_FS_Header::TimeSubSeconds`

File creation timestamp (sub-seconds)

Definition at line 216 of file `cfe_fs_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h`

11.74 CFE_SB_AllSubscriptionsTlm Struct Reference

`#include <cfe_sb_msg.h>`

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_SB_AllSubscriptionsTlm_Payload_t Payload`
Telemetry payload.

11.74.1 Detailed Description

Definition at line 797 of file cfe_sb_msg.h.

11.74.2 Field Documentation

11.74.2.1 Payload

`CFE_SB_AllSubscriptionsTlm_Payload_t` CFE_SB_AllSubscriptionsTlm::Payload

Telemetry payload.

Definition at line 800 of file cfe_sb_msg.h.

11.74.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` CFE_SB_AllSubscriptionsTlm::TelemetryHeader

Telemetry header.

Definition at line 799 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.75 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- `uint32 PktSegment`
Pkt number(starts at 1) in the series.
- `uint32 TotalSegments`
Total number of pkts needed to complete the request.
- `uint32 Entries`
Number of entries in the pkt.
- `CFE_SB_SubEntries_t Entry [CFE_SB_SUB_ENTRIES_PER_PKT]`
Array of `CFE_SB_SubEntries_t` entries.

11.75.1 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 788 of file cfe_sb_msg.h.

11.75.2 Field Documentation

11.75.2.1 Entries

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload::Entries
```

Number of entries in the pkt.

Definition at line 793 of file cfe_sb_msg.h.

11.75.2.2 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 794 of file cfe_sb_msg.h.

11.75.2.3 PktSegment

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload::PktSegment
```

Pkt number(starts at 1) in the series.

Definition at line 791 of file cfe_sb_msg.h.

11.75.2.4 TotalSegments

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload::TotalSegments
```

Total number of pkts needed to complete the request.

Definition at line 792 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.76 CFE_SB_HousekeepingTlm Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- **CFE_MSG_TelemetryHeader_t TelemetryHeader**
Telemetry header.
- **CFE_SB_HousekeepingTlm_Payload_t Payload**
Telemetry payload.

11.76.1 Detailed Description

Definition at line 598 of file cfe_sb_msg.h.

11.76.2 Field Documentation

11.76.2.1 Payload

```
CFE_SB_HousekeepingTlm_Payload_t CFE_SB_HousekeepingTlm::Payload
```

Telemetry payload.

Definition at line 601 of file cfe_sb_msg.h.

11.76.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` CFE_SB_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 600 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/fsw/inc/cfe_sb_msg.h](#)

11.77 CFE_SB_HousekeepingTlm_Payload Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- `uint8 CommandCounter`
Count of valid commands received.
- `uint8 CommandErrorCounter`
Count of invalid commands received.
- `uint8 NoSubscribersCounter`
Count pkts sent with no subscribers.
- `uint8 MsgSendErrorCounter`
Count of message send errors.
- `uint8 MsgReceiveErrorCounter`
Count of message receive errors.
- `uint8 InternalErrorCounter`
Count of queue read or write errors.
- `uint8 CreatePipeErrorCounter`
Count of errors in create pipe API.
- `uint8 SubscribeErrorCounter`
Count of errors in subscribe API.
- `uint8 PipeOptsErrorCounter`
Count of errors in set/get pipe options API.
- `uint8 DuplicateSubscriptionsCounter`
Count of duplicate subscriptions.
- `uint8 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

11.77.1 Detailed Description

Name Software Bus task housekeeping Packet

Definition at line 554 of file cfe_sb_msg.h.

11.77.2 Field Documentation

11.77.2.1 CommandCounter

`uint8 CFE_SB_HousekeepingTlm_Payload::CommandCounter`

Count of valid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_CMDPC

Definition at line 557 of file cfe_sb_msg.h.

11.77.2.2 CommandErrorCounter

`uint8 CFE_SB_HousekeepingTlm_Payload::CommandErrorCounter`

Count of invalid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_CMDEC

Definition at line 559 of file cfe_sb_msg.h.

11.77.2.3 CreatePipeErrorCounter

`uint8 CFE_SB_HousekeepingTlm_Payload::CreatePipeErrorCounter`

Count of errors in create pipe API.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_NewPipeEC

Definition at line 570 of file cfe_sb_msg.h.

11.77.2.4 DuplicateSubscriptionsCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptionsCounter
```

Count of duplicate subscriptions.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_DupSubCnt

Definition at line 576 of file cfe_sb_msg.h.

11.77.2.5 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 578 of file cfe_sb_msg.h.

11.77.2.6 InternalErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::InternalErrorCounter
```

Count of queue read or write errors.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_InternalEC

Definition at line 568 of file cfe_sb_msg.h.

11.77.2.7 MemInUse

```
uint32 CFE_SB_HousekeepingTlm_Payload::MemInUse
```

Memory in use.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_MemInUse

Definition at line 591 of file cfe_sb_msg.h.

11.77.2.8 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 588 of file cfe_sb_msg.h.

11.77.2.9 MsgLimitErrorCounter

```
uint16 CFE_SB_HousekeepingTlm_Payload::MsgLimitErrorCounter
```

Count of msg id to pipe errors.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_MsgLimEC

Definition at line 585 of file cfe_sb_msg.h.

11.77.2.10 MsgReceiveErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::MsgReceiveErrorCounter
```

Count of message receive errors.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_MsgRecEC

Definition at line 566 of file cfe_sb_msg.h.

11.77.2.11 MsgSendErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::MsgSendErrorCounter
```

Count of message send errors.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_MsgSndEC

Definition at line 563 of file cfe_sb_msg.h.

11.77.2.12 NoSubscribersCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::NoSubscribersCounter
```

Count pkts sent with no subscribers.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_NoSubEC

Definition at line 561 of file cfe_sb_msg.h.

11.77.2.13 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 574 of file cfe_sb_msg.h.

11.77.2.14 PipeOverflowErrorCounter

```
uint16 CFE_SB_HousekeepingTlm_Payload::PipeOverflowErrorCounter
```

Count of pipe overflow errors.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_PipeOvrEC

Definition at line 583 of file cfe_sb_msg.h.

11.77.2.15 Spare2Align

```
uint8 CFE_SB_HousekeepingTlm_Payload::Spare2Align[1]
```

Spare bytes to ensure alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Spare2Align[2]

Definition at line 580 of file cfe_sb_msg.h.

11.77.2.16 SubscribeErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload::SubscribeErrorCounter
```

Count of errors in subscribe API.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_SubscrEC

Definition at line 572 of file cfe_sb_msg.h.

11.77.2.17 UnmarkedMem

```
uint32 CFE_SB_HousekeepingTlm_Payload::UnmarkedMem
```

cfg param CFE_PLATFORM_SB_BUF_MEMORY_BYTES minus Peak Memory in use

Telemetry Mnemonic(s) \$sc_\$cpu_SB_UnMarkedMem

Definition at line 594 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.78 CFE_SB_Msg Union Reference

Software Bus generic message.

```
#include <cfe_sb_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.

11.78.1 Detailed Description

Software Bus generic message.

Definition at line 142 of file cfe_sb_api_typedefs.h.

11.78.2 Field Documentation

11.78.2.1 LongDouble

```
long double CFE_SB_Msg::LongDouble
```

Align to support Long Double.

Definition at line 146 of file `cfe_sb_api_typedefs.h`.

11.78.2.2 LongInt

```
long long int CFE_SB_Msg::LongInt
```

Align to support Long Integer.

Definition at line 145 of file `cfe_sb_api_typedefs.h`.

11.78.2.3 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 `HS_AppPipe()`, `HS_CustomCommands()`, `HS_DisableAlivenessCmd()`, `HS_DisableAppMonCmd()`, `HS_DisableCPUHogCmd()`, `HS_DisableEventMonCmd()`, `HS_EnableAlivenessCmd()`, `HS_EnableAppMonCmd()`, `HS_EnableCPUHogCmd()`, `HS_EnableEventMonCmd()`, `HS_HousekeepingReq()`, `HS_MonitorApplications()`, `HS_MonitorEvent()`, `HS_NoopCmd()`, `HS_ResetCmd()`, `HS_ResetResetsPerformedCmd()`, `HS_SetMaxResetsCmd()`, `HS_SetUtilDiagCmd()`, `HS_SetUtilParamsCmd()`, and `HS_ValidateMATable()`.

The documentation for this union was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h`

11.79 CFE_SB_MsgId_t Struct Reference

[CFE_SB_MsgId_t](#) type definition.

```
#include <cfe_sb_extern_typedefs.h>
```

Data Fields

- [CFE_SB_MsgId_Atom_t Value](#)

11.79.1 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 116 of file [cfe_sb_extern_typedefs.h](#).

11.79.2 Field Documentation**11.79.2.1 Value**

[CFE_SB_MsgId_Atom_t](#) [CFE_SB_MsgId_t::Value](#)

Definition at line 118 of file [cfe_sb_extern_typedefs.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h](#)

11.80 CFE_SB_MsgMapFileEntry Struct Reference

SB Map File Entry.

```
#include <cfe_sb_msg.h>
```

Data Fields

- [CFE_SB_MsgId_t MsgId](#)
Message Id which has been subscribed to.
- [CFE_SB_Routeld_Atom_t Index](#)
Routing raw index value (0 based, not Route ID)

11.80.1 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 730 of file cfe_sb_msg.h.

11.80.2 Field Documentation

11.80.2.1 Index

[CFE_SB_RouteId_Atom_t](#) CFE_SB_MsgMapFileEntry::Index

Routing raw index value (0 based, not Route ID)

Definition at line 733 of file cfe_sb_msg.h.

11.80.2.2 MsgId

[CFE_SB_MsgId_t](#) CFE_SB_MsgMapFileEntry::MsgId

Message Id which has been subscribed to.

Definition at line 732 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfe_sb_msg.h](#)

11.81 CFE_SB_PipeDepthStats Struct Reference

SB Pipe Depth Statistics.

```
#include <cfe_sb_msg.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.

11.81.1 Detailed Description

SB Pipe Depth Statistics.

Used in SB Statistics Telemetry Packet [CFE_SB_StatsTlm_t](#)

Definition at line 609 of file cfe_sb_msg.h.

11.81.2 Field Documentation

11.81.2.1 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 616 of file cfe_sb_msg.h.

11.81.2.2 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 614 of file cfe_sb_msg.h.

11.81.2.3 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 618 of file cfe_sb_msg.h.

11.81.2.4 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 612 of file cfe_sb_msg.h.

11.81.2.5 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 620 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.82 CFE_SB_PipeInfoEntry Struct Reference

SB Pipe Information File Entry.

```
#include <cfe_sb_msg.h>
```

Data Fields

- `CFE_SB_PipeId_t PipeId`
- `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]`

11.82.1 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 640 of file cfe_sb_msg.h.

11.82.2 Field Documentation

11.82.2.1 AppId

`CFE_ES_AppId_t CFE_SB_PipeInfoEntry::AppId`

The runtime ID of the application that owns the pipe

Definition at line 643 of file cfe_sb_msg.h.

11.82.2.2 AppName

`char CFE_SB_PipeInfoEntry::AppName [CFE_MISSION_MAX_API_LEN]`

The Name of the application that owns the pipe

Definition at line 645 of file cfe_sb_msg.h.

11.82.2.3 CurrentQueueDepth

`uint16 CFE_SB_PipeInfoEntry::CurrentQueueDepth`

The current depth of the pipe

Definition at line 647 of file `cfe_sb_msg.h`.

11.82.2.4 MaxQueueDepth

`uint16 CFE_SB_PipeInfoEntry::MaxQueueDepth`

The allocated depth of the pipe (max capacity)

Definition at line 646 of file `cfe_sb_msg.h`.

11.82.2.5 Opts

`uint8 CFE_SB_PipeInfoEntry::Opts`

Pipe options set (bitmask)

Definition at line 650 of file `cfe_sb_msg.h`.

11.82.2.6 PeakQueueDepth

`uint16 CFE_SB_PipeInfoEntry::PeakQueueDepth`

The peak depth of the pipe (high watermark)

Definition at line 648 of file `cfe_sb_msg.h`.

11.82.2.7 Pipelid

`CFE_SB_PipeId_t CFE_SB_PipeInfoEntry::PipeId`

The runtime ID of the pipe

Definition at line 642 of file `cfe_sb_msg.h`.

11.82.2.8 PipeName

```
char CFE_SB_PipeInfoEntry::PipeName[CFE_MISSION_MAX_API_LEN]
```

The Name of the pipe

Definition at line 644 of file cfe_sb_msg.h.

11.82.2.9 SendErrors

```
uint16 CFE_SB_PipeInfoEntry::SendErrors
```

Number of errors when writing to this pipe

Definition at line 649 of file cfe_sb_msg.h.

11.82.2.10 Spare

```
uint8 CFE_SB_PipeInfoEntry::Spare[3]
```

Padding to make this structure a multiple of 4 bytes

Definition at line 651 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.83 CFE_SB_Qos_t Struct Reference

Quality Of Service Type Definition.

```
#include <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.

11.83.1 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 133 of file cfe_sb_extern_typedefs.h.

11.83.2 Field Documentation

11.83.2.1 Priority

`uint8 CFE_SB_Qos_t::Priority`

Specify high(1) or low(0) message priority for off-board routing, currently unused.

Definition at line 135 of file cfe_sb_extern_typedefs.h.

11.83.2.2 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 136 of file cfe_sb_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h](#)

11.84 CFE_SB_RouteCmd Struct Reference

Enable/Disable Route Command.

```
#include <cfe_sb_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_SB_RouteCmd_Payload_t Payload](#)
Command payload.

11.84.1 Detailed Description

Enable/Disable Route Command.

Definition at line 535 of file cfe_sb_msg.h.

11.84.2 Field Documentation

11.84.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_SB_RouteCmd::CommandHeader

Command header.

Definition at line 537 of file cfe_sb_msg.h.

11.84.2.2 Payload

[CFE_SB_RouteCmd_Payload_t](#) CFE_SB_RouteCmd::Payload

Command payload.

Definition at line 538 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfe_sb_msg.h](#)

11.85 CFE_SB_RouteCmd_Payload Struct Reference

Enable/Disable Route Command Payload.

```
#include <cfe_sb_msg.h>
```

Data Fields

- [CFE_SB_MsgId_t](#) MsgId
Message ID of route to be enabled or disabled [CFE_SB_MsgId_t](#).
- [CFE_SB_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.

11.85.1 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 PipeId combination.

Definition at line 524 of file `cfe_sb_msg.h`.

11.85.2 Field Documentation

11.85.2.1 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 527 of file `cfe_sb_msg.h`.

11.85.2.2 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 528 of file `cfe_sb_msg.h`.

11.85.2.3 Spare

`uint8` `CFE_SB_RouteCmd_Payload::Spare`

Spare byte to make command even number of bytes.

Definition at line 529 of file `cfe_sb_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/fsw/inc/cfe_sb_msg.h`

11.86 CFE_SB_RoutingFileEntry Struct Reference

SB Routing File Entry.

```
#include <cfе_sb_msg.h>
```

Data Fields

- **CFE_SB_MsgId_t MsgId**
Message Id portion of the route.
- **CFE_SB_Pipeld_t Pipeld**
Pipe Id portion of the route.
- **uint8 State**
Route Enabled or Disabled.
- **uint16 MsgCnt**
Number of msgs with this MsgId sent to this Pipeld.
- **char AppName [CFE_MISSION_MAX_API_LEN]**
Pipe Depth Statistics.
- **char PipeName [CFE_MISSION_MAX_API_LEN]**
Pipe Depth Statistics.

11.86.1 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 715 of file cfe_sb_msg.h.

11.86.2 Field Documentation

11.86.2.1 AppName

```
char CFE_SB_RoutingFileEntry::AppName[CFE_MISSION_MAX_API_LEN]
```

Pipe Depth Statistics.

Definition at line 721 of file cfe_sb_msg.h.

11.86.2.2 MsgCnt

```
uint16 CFE_SB_RoutingFileEntry::MsgCnt
```

Number of msgs with this MsgId sent to this PipeId.

Definition at line 720 of file [cfe_sb_msg.h](#).

11.86.2.3 MsgId

```
CFE_SB_MsgId_t CFE_SB_RoutingFileEntry::MsgId
```

Message Id portion of the route.

Definition at line 717 of file [cfe_sb_msg.h](#).

11.86.2.4 PipeId

```
CFE_SB_PipeId_t CFE_SB_RoutingFileEntry::PipeId
```

Pipe Id portion of the route.

Definition at line 718 of file [cfe_sb_msg.h](#).

11.86.2.5 PipeName

```
char CFE_SB_RoutingFileEntry::PipeName[CFE_MISSION_MAX_API_LEN]
```

Pipe Depth Statistics.

Definition at line 722 of file [cfe_sb_msg.h](#).

11.86.2.6 State

```
uint8 CFE_SB_RoutingFileEntry::State
```

Route Enabled or Disabled.

Definition at line 719 of file [cfe_sb_msg.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/fsw/inc/cfe_sb_msg.h](#)

11.87 CFE_SB_SingleSubscriptionTlm Struct Reference

```
#include <cfе_sb_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_SB_SingleSubscriptionTlm_Payload_t Payload](#)
Telemetry payload.

11.87.1 Detailed Description

Definition at line 756 of file cfe_sb_msg.h.

11.87.2 Field Documentation

11.87.2.1 Payload

[CFE_SB_SingleSubscriptionTlm_Payload_t](#) CFE_SB_SingleSubscriptionTlm::Payload

Telemetry payload.

Definition at line 759 of file cfe_sb_msg.h.

11.87.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_SB_SingleSubscriptionTlm::TelemetryHeader

Telemetry header.

Definition at line 758 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfe_sb_msg.h](#)

11.88 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference

```
#include <cfе_sb_msg.h>
```

Data Fields

- [uint8 SubType](#)
Subscription or Unsubscription.
- [CFE_SB_MsgId_t MsgId](#)
MsgId subscribed or unsubscribe to.
- [CFE_SB_Qos_t Qos](#)
Quality of Service, used only for interprocessor communication.
- [CFE_SB_PipeId_t Pipe](#)
Destination pipe id to send above msg id.

11.88.1 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 746 of file cfe_sb_msg.h.

11.88.2 Field Documentation

11.88.2.1 MsgId

[CFE_SB_MsgId_t](#) CFE_SB_SingleSubscriptionTlm_Payload::MsgId

MsgId subscribed or unsubscribe to.

Definition at line 750 of file cfe_sb_msg.h.

11.88.2.2 Pipe

[CFE_SB_PipeId_t](#) CFE_SB_SingleSubscriptionTlm_Payload::Pipe

Destination pipe id to send above msg id.

Definition at line 752 of file cfe_sb_msg.h.

11.88.2.3 Qos

`CFE_SB_Qos_t CFE_SB_SingleSubscriptionTlm_Payload::Qos`

Quality of Service, used only for interprocessor communication.

Definition at line 751 of file `cfe_sb_msg.h`.

11.88.2.4 SubType

`uint8 CFE_SB_SingleSubscriptionTlm_Payload::SubType`

Subscription or Unsubscription.

Definition at line 749 of file `cfe_sb_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/fsw/inc/cfe_sb_msg.h`

11.89 CFE_SB_StatsTlm Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_SB_StatsTlm_Payload_t Payload`
Telemetry payload.

11.89.1 Detailed Description

Definition at line 704 of file `cfe_sb_msg.h`.

11.89.2 Field Documentation

11.89.2.1 Payload

`CFE_SB_StatsTlm_Payload_t` `CFE_SB_StatsTlm::Payload`

Telemetry payload.

Definition at line 707 of file `cfe_sb_msg.h`.

11.89.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` `CFE_SB_StatsTlm::TelemetryHeader`

Telemetry header.

Definition at line 706 of file `cfe_sb_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/fsw/inc/cfe_sb_msg.h`

11.90 CFE_SB_StatsTlm_Payload Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- **uint32 MsgIdsInUse**
Current number of MsgIds with a destination.
- **uint32 PeakMsgIdsInUse**
Peak number of MsgIds with a destination.
- **uint32 MaxMsgIdsAllowed**
cFE Cfg Param `CFE_PLATFORM_SB_MAX_MSG_IDS`
- **uint32 PipesInUse**
Number of pipes currently in use.
- **uint32 PeakPipesInUse**
Peak number of pipes since last reboot.
- **uint32 MaxPipesAllowed**
cFE Cfg Param `CFE_PLATFORM_SB_MAX_PIPES`
- **uint32 MemInUse**
Memory bytes currently in use for SB msg transfers.
- **uint32 PeakMemInUse**
Peak memory bytes in use for SB msg transfers.
- **uint32 MaxMemAllowed**
cFE Cfg Param `CFE_PLATFORM_SB_BUF_MEMORY_BYTES`
- **uint32 SubscriptionsInUse**

- `uint32 PeakSubscriptionsInUse`
Number of current subscriptions.
- `uint32 MaxSubscriptionsAllowed`
Peak number of subscriptions.
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`.

11.90.1 Detailed Description

Name SB Statistics Telemetry Packet

SB Statistics packet sent in response to `CFE_SB_SEND_SB_STATS_CC`

Definition at line 660 of file `cfe_sb_msg.h`.

11.90.2 Field Documentation

11.90.2.1 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 681 of file `cfe_sb_msg.h`.

11.90.2.2 MaxMsgIdsAllowed

`uint32 CFE_SB_StatsTlm_Payload::MaxMsgIdsAllowed`

cFE Cfg Param `CFE_PLATFORM_SB_MAX_MSG_IDS`

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMMMIDALW

Definition at line 667 of file `cfe_sb_msg.h`.

11.90.2.3 MaxPipeDepthAllowed

`uint32 CFE_SB_StatsTlm_Payload::MaxPipeDepthAllowed`

Maximum allowed pipe depth.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMMPDALW

Definition at line 697 of file cfe_sb_msg.h.

11.90.2.4 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 674 of file cfe_sb_msg.h.

11.90.2.5 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 688 of file cfe_sb_msg.h.

11.90.2.6 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 677 of file cfe_sb_msg.h.

11.90.2.7 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 663 of file cfe_sb_msg.h.

11.90.2.8 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 679 of file cfe_sb_msg.h.

11.90.2.9 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 665 of file cfe_sb_msg.h.

11.90.2.10 PeakPipesInUse

```
uint32 CFE_SB_StatsTlm_Payload::PeakPipesInUse
```

Peak number of pipes since last reboot.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPIU

Definition at line 672 of file cfe_sb_msg.h.

11.90.2.11 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 694 of file cfe_sb_msg.h.

11.90.2.12 PeakSubscriptionsInUse

`uint32 CFE_SB_StatsTlm_Payload::PeakSubscriptionsInUse`

Peak number of subscriptions.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMPSIU

Definition at line 686 of file cfe_sb_msg.h.

11.90.2.13 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 700 of file cfe_sb_msg.h.

11.90.2.14 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 670 of file cfe_sb_msg.h.

11.90.2.15 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 692 of file cfe_sb_msg.h.

11.90.2.16 SubscriptionsInUse

```
uint32 CFE_SB_StatsTlm_Payload::SubscriptionsInUse
```

Number of current subscriptions.

Telemetry Mnemonic(s) \$sc_\$cpu_SB_Stat.SB_SMSIU

Definition at line 684 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/cfe_sb_msg.h

11.91 CFE_SB_SubEntries Struct Reference

SB Previous Subscriptions Entry.

```
#include <cfe_sb_msg.h>
```

Data Fields

- **CFE_SB_MsgId_t MsgId**
MsgId portion of the subscription.
- **CFE_SB_Qos_t Qos**
Qos portion of the subscription.
- **CFE_SB_PipeId_t Pipe**
PipeId portion of the subscription.

11.91.1 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_AllSubscriptionsTim_t](#)

Definition at line 770 of file cfe_sb_msg.h.

11.91.2 Field Documentation

11.91.2.1 MsgId

[CFE_SB_MsgId_t](#) CFE_SB_SubEntries::MsgId

MsgId portion of the subscription.

Definition at line 773 of file cfe_sb_msg.h.

11.91.2.2 Pipe

[CFE_SB_PipeId_t](#) CFE_SB_SubEntries::Pipe

PipeId portion of the subscription.

Definition at line 775 of file cfe_sb_msg.h.

11.91.2.3 Qos

[CFE_SB_Qos_t](#) CFE_SB_SubEntries::Qos

Qos portion of the subscription.

Definition at line 774 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfe_sb_msg.h](#)

11.92 CFE_SB_WriteFileInfoCmd Struct Reference

Write File Info Command.

```
#include <cfе_sb_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_SB_WriteFileInfoCmd_Payload_t](#) `Payload`
Command payload.

11.92.1 Detailed Description

Write File Info Command.

Definition at line 503 of file cfe_sb_msg.h.

11.92.2 Field Documentation

11.92.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) `CFE_SB_WriteFileInfoCmd::CommandHeader`

Command header.

Definition at line 505 of file cfe_sb_msg.h.

11.92.2.2 Payload

[CFE_SB_WriteFileInfoCmd_Payload_t](#) `CFE_SB_WriteFileInfoCmd::Payload`

Command payload.

Definition at line 506 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfе_sb_msg.h](#)

11.93 CFE_SB_WriteFileInfoCmd_Payload Struct Reference

Write File Info Command Payload.

```
#include <cfе_sb_msg.h>
```

Data Fields

- char [Filename \[CFE_MISSION_MAX_PATH_LEN\]](#)

Path and Filename of data to be loaded.

11.93.1 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 495 of file cfe_sb_msg.h.

11.93.2 Field Documentation

11.93.2.1 Filename

```
char CFE_SB_WriteFileInfoCmd_Payload::Filename[CFE_MISSION_MAX_PATH_LEN]
```

Path and Filename of data to be loaded.

Definition at line 497 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/fsw/inc/[cfе_sb_msg.h](#)

11.94 CFE_TBL_AbortLoadCmd Struct Reference

Abort Load Command.

```
#include <cfе_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_AbortLoadCmd_Payload_t](#) Payload
Command payload.

11.94.1 Detailed Description

Abort Load Command.

Definition at line 684 of file cfe_tbl_msg.h.

11.94.2 Field Documentation

11.94.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_AbortLoadCmd::CommandHeader

Command header.

Definition at line 686 of file cfe_tbl_msg.h.

11.94.2.2 Payload

[CFE_TBL_AbortLoadCmd_Payload_t](#) CFE_TBL_AbortLoadCmd::Payload

Command payload.

Definition at line 687 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.95 CFE_TBL_AbortLoadCmd_Payload Struct Reference

Abort Load Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Full Name of Table whose load is to be aborted.

11.95.1 Detailed Description

Abort Load Command Payload.

For command details, see [CFE_TBL_ABORT_LOAD_CC](#)

Definition at line 674 of file cfe_tbl_msg.h.

11.95.2 Field Documentation

11.95.2.1 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 676 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h](#)

11.96 CFE_TBL_ActivateCmd Struct Reference

Activate Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TBL_ActivateCmd_Payload_t Payload](#)
Command payload.

11.96.1 Detailed Description

Activate Table Command.

Definition at line 595 of file cfe_tbl_msg.h.

11.96.2 Field Documentation

11.96.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_ActivateCmd::CommandHeader

Command header.

Definition at line 597 of file cfe_tbl_msg.h.

11.96.2.2 Payload

[CFE_TBL_ActivateCmd_Payload_t](#) CFE_TBL_ActivateCmd::Payload

Command payload.

Definition at line 598 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.97 CFE_TBL_ActivateCmd_Payload Struct Reference

Activate Table Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Full Name of Table to be activated.

11.97.1 Detailed Description

Activate Table Command Payload.

For command details, see [CFE_TBL_ACTIVATE_CC](#)

Definition at line 585 of file cfe_tbl_msg.h.

11.97.2 Field Documentation

11.97.2.1 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 587 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.98 CFE_TBL_DelCDSCmd_Payload Struct Reference

Delete Critical Table CDS Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)
Full Name of Table whose CDS is to be deleted.

11.98.1 Detailed Description

Delete Critical Table CDS Command Payload.

For command details, see [CFE_TBL_DELETE_CDS_CC](#)

Definition at line 651 of file cfe_tbl_msg.h.

11.98.2 Field Documentation

11.98.2.1 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 653 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.99 CFE_TBL_DeleteCDSCmd Struct Reference

Delete Critical Table CDS Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) `CommandHeader`
Command header.
- [CFE_TBL_DelCDSCmd_Payload_t](#) `Payload`
Command payload.

11.99.1 Detailed Description

Delete Critical Table CDS Command.

Definition at line 663 of file `cfe_tbl_msg.h`.

11.99.2 Field Documentation

11.99.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_DeleteCDSCmd::CommandHeader

Command header.

Definition at line 665 of file cfe_tbl_msg.h.

11.99.2.2 Payload

[CFE_TBL_DelCDSCmd_Payload_t](#) CFE_TBL_DeleteCDSCmd::Payload

Command payload.

Definition at line 666 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.100 CFE_TBL_DumpCmd Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_DumpCmd_Payload_t](#) Payload
Command payload.

11.100.1 Detailed Description

/brief Dump Table Command

Definition at line 547 of file cfe_tbl_msg.h.

11.100.2 Field Documentation

11.100.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_TBL_DumpCmd::CommandHeader`

Command header.

Definition at line 549 of file `cfe_tbl_msg.h`.

11.100.2.2 Payload

`CFE_TBL_DumpCmd_Payload_t` `CFE_TBL_DumpCmd::Payload`

Command payload.

Definition at line 550 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.101 CFE_TBL_DumpCmd_Payload Struct Reference

Dump Table Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- `uint16 ActiveTableFlag`
CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table
- `char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`
Full name of table to be dumped.
- `char DumpFilename [CFE_MISSION_MAX_PATH_LEN]`
Full Filename where data is to be written.

11.101.1 Detailed Description

Dump Table Command Payload.

For command details, see [CFE_TBL_DUMP_CC](#)

Definition at line 528 of file `cfe_tbl_msg.h`.

11.101.2 Field Documentation

11.101.2.1 ActiveTableFlag

`uint16 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 530 of file `cfe_tbl_msg.h`.

11.101.2.2 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 539 of file `cfe_tbl_msg.h`.

11.101.2.3 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 536 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.102 CFE_TBL_DumpRegistryCmd Struct Reference

Dump Registry Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_DumpRegistryCmd_Payload_t](#) Payload
Command payload.

11.102.1 Detailed Description

Dump Registry Command.

Definition at line 617 of file cfe_tbl_msg.h.

11.102.2 Field Documentation

11.102.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_DumpRegistryCmd::CommandHeader

Command header.

Definition at line 619 of file cfe_tbl_msg.h.

11.102.2.2 Payload

[CFE_TBL_DumpRegistryCmd_Payload_t](#) CFE_TBL_DumpRegistryCmd::Payload

Command payload.

Definition at line 620 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.103 CFE_TBL_DumpRegistryCmd_Payload Struct Reference

Dump Registry Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char [DumpFilename \[CFE_MISSION_MAX_PATH_LEN\]](#)

Full Filename where dumped data is to be written.

11.103.1 Detailed Description

Dump Registry Command Payload.

For command details, see [CFE_TBL_DUMP_REGISTRY_CC](#)

Definition at line 606 of file cfe_tbl_msg.h.

11.103.2 Field Documentation

11.103.2.1 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 608 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h](#)

11.104 CFE_TBL_File_Hdr Struct Reference

The definition of the header fields that are included in CFE Table Data files.

```
#include <cfe_tbl_extern_typedefs.h>
```

Data Fields

- [uint32 Reserved](#)
- [CFE_ES_MemOffset_t Offset](#)
- [CFE_ES_MemOffset_t NumBytes](#)
- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

11.104.1 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.

Definition at line 71 of file `cfe_tbl_extern_typedefs.h`.

11.104.2 Field Documentation

11.104.2.1 NumBytes

`CFE_ES_MemOffset_t` `CFE_TBL_File_Hdr::NumBytes`

Number of bytes to load into table

Definition at line 75 of file `cfe_tbl_extern_typedefs.h`.

11.104.2.2 Offset

`CFE_ES_MemOffset_t` `CFE_TBL_File_Hdr::Offset`

Byte Offset at which load should commence

Definition at line 74 of file `cfe_tbl_extern_typedefs.h`.

11.104.2.3 Reserved

`uint32` `CFE_TBL_File_Hdr::Reserved`

Future Use: NumTblSegments in File?

Definition at line 73 of file `cfe_tbl_extern_typedefs.h`.

11.104.2.4 TableName

`char` `CFE_TBL_File_Hdr::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

Fully qualified name of table to load

Definition at line 76 of file `cfe_tbl_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h`

11.105 CFE_TBL_FileDef Struct Reference

```
#include <cfef_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.

11.105.1 Detailed Description

Definition at line 56 of file `cfef_tbl_filedef.h`.

11.105.2 Field Documentation

11.105.2.1 Description

```
char CFE_TBL_FileDef::Description[CFE\_FS\_HDR\_DESC\_MAX\_LEN]
```

Description of table image that is included in cFE File Header.

Definition at line 60 of file `cfef_tbl_filedef.h`.

11.105.2.2 ObjectName

```
char CFE_TBL_FileDef::ObjectName[ 64 ]
```

Name of instantiated variable that contains desired table image.

Definition at line 58 of file `cfef_tbl_filedef.h`.

11.105.2.3 ObjectSize

```
uint32 CFE_TBL_FileDef::ObjectSize
```

Size, in bytes, of instantiated object.

Definition at line 64 of file cfe_tbl_filedef.h.

11.105.2.4 TableName

```
char CFE_TBL_FileDef::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Name of Table as defined onboard.

Definition at line 59 of file cfe_tbl_filedef.h.

11.105.2.5 TgtFilename

```
char CFE_TBL_FileDef::TgtFilename[CFE_MISSION_MAX_FILE_LEN]
```

Default filename to be used for output of elf2cfetbl utility.

Definition at line 62 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

11.106 CFE_TBL_HousekeepingTlm Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_TBL_HousekeepingTlm_Payload_t Payload](#)
Telemetry payload.

11.106.1 Detailed Description

Definition at line 783 of file cfe_tbl_msg.h.

11.106.2 Field Documentation

11.106.2.1 Payload

`CFE_TBL_HousekeepingTlm_Payload_t` CFE_TBL_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 786 of file cfe_tbl_msg.h.

11.106.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` CFE_TBL_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 785 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h

11.107 CFE_TBL_HousekeepingTlm_Payload Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- `uint8 CommandCounter`
Count of valid commands received.
- `uint8 CommandErrorCounter`
Count of invalid commands received.
- `uint16 NumTables`
Number of Tables Registered.
- `uint16 NumLoadPending`
Number of Tables pending on Applications for their update.
- `uint16 ValidationCounter`
Number of completed table validations.
- `uint32 LastValCrc`
Data Integrity Value computed for last table validated.
- `int32 LastValStatus`
Returned status from validation function for last table validated.
- `bool ActiveBuffer`

- `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.

11.107.1 Detailed Description

Name Table Services Housekeeping Packet

Definition at line 724 of file cfe_tbl_msg.h.

11.107.2 Field Documentation

11.107.2.1 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 751 of file cfe_tbl_msg.h.

11.107.2.2 ByteAlignPad1

```
uint8 CFE_TBL_HousekeepingTlm_Payload::ByteAlignPad1
```

Spare byte to ensure longword alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ByteAlignPad1

Definition at line 767 of file cfe_tbl_msg.h.

11.107.2.3 CommandCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload::CommandCounter
```

Count of valid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CMDPC

Definition at line 729 of file cfe_tbl_msg.h.

11.107.2.4 CommandErrorCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload::CommandErrorCounter
```

Count of invalid commands received.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CMDEC

Definition at line 731 of file cfe_tbl_msg.h.

11.107.2.5 FailedValCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload::FailedValCounter
```

Total number of unsuccessful table validations.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ValFailedCtr

Definition at line 757 of file cfe_tbl_msg.h.

11.107.2.6 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 777 of file cfe_tbl_msg.h.

11.107.2.7 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 775 of file cfe_tbl_msg.h.

11.107.2.8 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 779 of file cfe_tbl_msg.h.

11.107.2.9 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 773 of file cfe_tbl_msg.h.

11.107.2.10 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 771 of file cfe_tbl_msg.h.

11.107.2.11 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 747 of file cfe_tbl_msg.h.

11.107.2.12 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 749 of file cfe_tbl_msg.h.

11.107.2.13 LastValTableName

```
char CFE_TBL_HousekeepingTlm_Payload::LastValTableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Name of last table validated.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]

Definition at line 753 of file cfe_tbl_msg.h.

11.107.2.14 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 769 of file cfe_tbl_msg.h.

11.107.2.15 NumFreeSharedBufs

`uint8` `CFE_TBL_HousekeepingTlm_Payload::NumFreeSharedBufs`

Number of free Shared Working Buffers.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_NumFreeShrBuf

Definition at line 765 of file cfe_tbl_msg.h.

11.107.2.16 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 739 of file cfe_tbl_msg.h.

11.107.2.17 NumTables

`uint16` `CFE_TBL_HousekeepingTlm_Payload::NumTables`

Number of Tables Registered.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_NumTables

Definition at line 737 of file cfe_tbl_msg.h.

11.107.2.18 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 759 of file cfe_tbl_msg.h.

11.107.2.19 SuccessValCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload::SuccessValCounter
```

Total number of successful table validations.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ValSuccessCtr

Definition at line 755 of file cfe_tbl_msg.h.

11.107.2.20 ValidationCounter

```
uint16 CFE_TBL_HousekeepingTlm_Payload::ValidationCounter
```

Number of completed table validations.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ValCompltdCtr

Definition at line 745 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h

11.108 CFE_TBL_Info Struct Reference

Table Info.

```
#include <cfe_tbl_api_typedefs.h>
```

Data Fields

- `size_t Size`
Size, in bytes, of Table.
- `uint32 NumUsers`
Number of Apps with access to the table.
- `uint32 FileCreateTimeSecs`
File creation time from last file loaded into table.
- `uint32 FileCreateTimeSubSecs`
File creation time from last file loaded into table.
- `uint32 Crc`
Most recently calculated CRC by TBL services on table contents.
- `CFE_TIME_SysTime_t TimeOfLastUpdate`
Time when Table was last updated.
- `bool TableLoadedOnce`
Flag indicating whether table has been loaded once or not.
- `bool DumpOnly`
Flag indicating Table is NOT to be loaded.
- `bool DoubleBuffered`
Flag indicating Table has a dedicated inactive buffer.
- `bool UserDefAddr`
Flag indicating Table address was defined by Owner Application.
- `bool Critical`
Flag indicating Table contents are maintained in a CDS.
- `char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]`
Filename of last file loaded into table.

11.108.1 Detailed Description

Table Info.

Definition at line 109 of file `cfe_tbl_api_typedefs.h`.

11.108.2 Field Documentation

11.108.2.1 Crc

`uint32 CFE_TBL_Info::Crc`

Most recently calculated CRC by TBL services on table contents.

Definition at line 115 of file `cfe_tbl_api_typedefs.h`.

11.108.2.2 Critical

```
bool CFE_TBL_Info::Critical
```

Flag indicating Table contents are maintained in a CDS.

Definition at line 121 of file cfe_tbl_api_typedefs.h.

11.108.2.3 DoubleBuffered

```
bool CFE_TBL_Info::DoubleBuffered
```

Flag indicating Table has a dedicated inactive buffer.

Definition at line 119 of file cfe_tbl_api_typedefs.h.

11.108.2.4 DumpOnly

```
bool CFE_TBL_Info::DumpOnly
```

Flag indicating Table is NOT to be loaded.

Definition at line 118 of file cfe_tbl_api_typedefs.h.

11.108.2.5 FileCreateTimeSecs

```
uint32 CFE_TBL_Info::FileCreateTimeSecs
```

File creation time from last file loaded into table.

Definition at line 113 of file cfe_tbl_api_typedefs.h.

11.108.2.6 FileCreateTimeSubSecs

```
uint32 CFE_TBL_Info::FileCreateTimeSubSecs
```

File creation time from last file loaded into table.

Definition at line 114 of file cfe_tbl_api_typedefs.h.

11.108.2.7 LastFileLoaded

```
char CFE_TBL_Info::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]
```

Filename of last file loaded into table.

Definition at line 122 of file cfe_tbl_api_typedefs.h.

11.108.2.8 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.

11.108.2.9 Size

```
size_t CFE_TBL_Info::Size
```

Size, in bytes, of Table.

Definition at line 111 of file cfe_tbl_api_typedefs.h.

11.108.2.10 TableLoadedOnce

```
bool CFE_TBL_Info::TableLoadedOnce
```

Flag indicating whether table has been loaded once or not.

Definition at line 117 of file cfe_tbl_api_typedefs.h.

11.108.2.11 TimeOfLastUpdate

```
CFE_TIME_SysTime_t CFE_TBL_Info::TimeOfLastUpdate
```

Time when Table was last updated.

Definition at line 116 of file cfe_tbl_api_typedefs.h.

11.108.2.12 UserDefAddr

```
bool CFE_TBL_Info::UserDefAddr
```

Flag indicating Table address was defined by Owner Application.

Definition at line 120 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](#)

11.109 CFE_TBL_LoadCmd Struct Reference

Load Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TBL_LoadCmd_Payload_t Payload](#)
Command payload.

11.109.1 Detailed Description

Load Table Command.

Definition at line 517 of file `cfe_tbl_msg.h`.

11.109.2 Field Documentation

11.109.2.1 CommandHeader

```
CFE\_MSG\_CommandHeader\_t CFE_TBL_LoadCmd::CommandHeader
```

Command header.

Definition at line 519 of file `cfe_tbl_msg.h`.

11.109.2.2 Payload

`CFE_TBL_LoadCmd_Payload_t` `CFE_TBL_LoadCmd::Payload`

Command payload.

Definition at line 520 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.110 CFE_TBL_LoadCmd_Payload Struct Reference

Load Table Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- `char LoadFilename [CFE_MISSION_MAX_PATH_LEN]`
Filename (and path) of data to be loaded.

11.110.1 Detailed Description

Load Table Command Payload.

For command details, see [CFE_TBL_LOAD_CC](#)

Definition at line 509 of file `cfe_tbl_msg.h`.

11.110.2 Field Documentation

11.110.2.1 LoadFilename

```
char CFE_TBL_LoadCmd_Payload::LoadFilename [CFE_MISSION_MAX_PATH_LEN]
```

Filename (and path) of data to be loaded.

Definition at line 511 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.111 CFE_TBL_NoArgsCmd Struct Reference

Generic "no arguments" command.

```
#include <cfef_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.

11.111.1 Detailed Description

Generic "no arguments" command.

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see [CFE_TBL_NOOP_CC](#))
3. The Reset Counters Command (For details, see [CFE_TBL_RESET_COUNTERS_CC](#))

Definition at line 490 of file cfe_tbl_msg.h.

11.111.2 Field Documentation

11.111.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_NoArgsCmd::CommandHeader

Command header.

Definition at line 492 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.112 CFE_TBL_NotifyCmd Struct Reference

```
#include <cfef_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_NotifyCmd_Payload_t](#) Payload
Command payload.

11.112.1 Detailed Description

/brief Table Management Notification Command

Definition at line 711 of file cfe_tbl_msg.h.

11.112.2 Field Documentation

11.112.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_NotifyCmd::CommandHeader

Command header.

Definition at line 713 of file cfe_tbl_msg.h.

11.112.2.2 Payload

[CFE_TBL_NotifyCmd_Payload_t](#) CFE_TBL_NotifyCmd::Payload

Command payload.

Definition at line 714 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.113 CFE_TBL_NotifyCmd_Payload Struct Reference

Table Management Notification Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [uint32 Parameter](#)

Application specified command parameter.

11.113.1 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 703 of file `cfe_tbl_msg.h`.

11.113.2 Field Documentation

11.113.2.1 Parameter

`uint32 CFE_TBL_NotifyCmd_Payload::Parameter`

Application specified command parameter.

Definition at line 705 of file `cfe_tbl_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h`

11.114 CFE_TBL_SendRegistryCmd Struct Reference

Send Table Registry Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TBL_SendRegistryCmd_Payload_t Payload](#)
Command payload.

11.114.1 Detailed Description

Send Table Registry Command.

Definition at line 640 of file cfe_tbl_msg.h.

11.114.2 Field Documentation

11.114.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_SendRegistryCmd::CommandHeader

Command header.

Definition at line 642 of file cfe_tbl_msg.h.

11.114.2.2 Payload

[CFE_TBL_SendRegistryCmd_Payload_t](#) CFE_TBL_SendRegistryCmd::Payload

Command payload.

Definition at line 643 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.115 CFE_TBL_SendRegistryCmd_Payload Struct Reference

Send Table Registry Command Payload.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char [TableName \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Full Name of Table whose registry entry is to be telemetered.

11.115.1 Detailed Description

Send Table Registry Command Payload.

For command details, see [CFE_TBL_SEND_REGISTRY_CC](#)

Definition at line 628 of file cfe_tbl_msg.h.

11.115.2 Field Documentation

11.115.2.1 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 630 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfe_tbl_msg.h](#)

11.116 CFE_TBL_TableRegistryTlm Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_TBL_TblRegPacket_Payload_t Payload](#)
Telemetry payload.

11.116.1 Detailed Description

Definition at line 830 of file cfe_tbl_msg.h.

11.116.2 Field Documentation

11.116.2.1 Payload

`CFE_TBL_TblRegPacket_Payload_t` CFE_TBL_TableRegistryTlm::Payload

Telemetry payload.

Definition at line 833 of file cfe_tbl_msg.h.

11.116.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` CFE_TBL_TableRegistryTlm::TelemetryHeader

Telemetry header.

Definition at line 832 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h

11.117 CFE_TBL_TblRegPacket_Payload Struct Reference

```
#include <cfe_tbl_msg.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.
- `uint32` FileCreateTimeSecs
File creation time from last file loaded into table.
- `uint32` FileCreateTimeSubSecs

File creation time from last file loaded into table.

- bool [TableLoadedOnce](#)

Flag indicating whether table has been loaded once or not.

- bool [LoadPending](#)

Flag indicating an inactive buffer is ready to be copied.

- bool [DumpOnly](#)

Flag indicating Table is NOT to be loaded.

- bool [DoubleBuffered](#)

Flag indicating Table has a dedicated inactive buffer.

- char [Name \[CFE_MISSION_TBL_MAX_FULL_NAME_LEN\]](#)

Processor specific table name.

- char [LastFileLoaded \[CFE_MISSION_MAX_PATH_LEN\]](#)

Filename of last file loaded into table.

- char [OwnerAppName \[CFE_MISSION_MAX_API_LEN\]](#)

Name of owning application.

- bool [Critical](#)

Indicates whether table is Critical or not.

- uint8 [ByteAlign4](#)

Spare byte to maintain byte alignment.

11.117.1 Detailed Description

Name Table Registry Info Packet

Definition at line 792 of file cfe_tbl_msg.h.

11.117.2 Field Documentation

11.117.2.1 ActiveBufferAddr

[CFE_ES_MemAddress_t](#) [CFE_TBL_TblRegPacket_Payload::ActiveBufferAddr](#)

Address of Active Buffer.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_ActBufAdd

Definition at line 798 of file cfe_tbl_msg.h.

11.117.2.2 ByteAlign4

```
uint8 CFE_TBL_TblRegPacket_Payload::ByteAlign4
```

Spare byte to maintain byte alignment.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_Spare4

Definition at line 826 of file cfe_tbl_msg.h.

11.117.2.3 Crc

```
uint32 CFE_TBL_TblRegPacket_Payload::Crc
```

Most recently calculated CRC of Table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_CRC

Definition at line 796 of file cfe_tbl_msg.h.

11.117.2.4 Critical

```
bool CFE_TBL_TblRegPacket_Payload::Critical
```

Indicates whether table is Critical or not.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_Spare3

Definition at line 824 of file cfe_tbl_msg.h.

11.117.2.5 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 816 of file cfe_tbl_msg.h.

11.117.2.6 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 814 of file cfe_tbl_msg.h.

11.117.2.7 FileCreateTimeSecs

```
uint32 CFE_TBL_TblRegPacket_Payload::FileCreateTimeSecs
```

File creation time from last file loaded into table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_FILECSECONDS

Definition at line 806 of file cfe_tbl_msg.h.

11.117.2.8 FileCreateTimeSubSecs

```
uint32 CFE_TBL_TblRegPacket_Payload::FileCreateTimeSubSecs
```

File creation time from last file loaded into table.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_FILECSUBSECONDS

Definition at line 808 of file cfe_tbl_msg.h.

11.117.2.9 InactiveBufferAddr

```
CFE_ES_MemAddress_t CFE_TBL_TblRegPacket_Payload::InactiveBufferAddr
```

Address of Inactive Buffer.

Telemetry Mnemonic(s) \$sc_\$cpu_TBL_IActBufAdd

Definition at line 800 of file cfe_tbl_msg.h.

11.117.2.10 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 820 of file cfe_tbl_msg.h.

11.117.2.11 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 812 of file cfe_tbl_msg.h.

11.117.2.12 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 818 of file cfe_tbl_msg.h.

11.117.2.13 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 822 of file cfe_tbl_msg.h.

11.117.2.14 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 794 of file cfe_tbl_msg.h.

11.117.2.15 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 810 of file cfe_tbl_msg.h.

11.117.2.16 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 804 of file cfe_tbl_msg.h.

11.117.2.17 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 802 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h

11.118 CFE_TBL_ValidateCmd Struct Reference

Validate Table Command.

```
#include <cfef_tbl_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t](#) CommandHeader
Command header.
- [CFE_TBL_ValidateCmd_Payload_t](#) Payload
Command payload.

11.118.1 Detailed Description

Validate Table Command.

Definition at line 574 of file cfe_tbl_msg.h.

11.118.2 Field Documentation

11.118.2.1 CommandHeader

[CFE_MSG_CommandHeader_t](#) CFE_TBL_ValidateCmd::CommandHeader

Command header.

Definition at line 576 of file cfe_tbl_msg.h.

11.118.2.2 Payload

[CFE_TBL_ValidateCmd_Payload_t](#) CFE_TBL_ValidateCmd::Payload

Command payload.

Definition at line 577 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/[cfef_tbl_msg.h](#)

11.119 CFE_TBL_ValidateCmd_Payload Struct Reference

Validate Table Command Payload.

```
#include <cfef_tbl_msg.h>
```

Data Fields

- `uint16 ActiveTableFlag`
CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table
- `char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`
Full Name of Table to be validated.

11.119.1 Detailed Description

Validate Table Command Payload.

For command details, see [CFE_TBL_VALIDATE_CC](#)

Definition at line 558 of file cfe_tbl_msg.h.

11.119.2 Field Documentation

11.119.2.1 ActiveTableFlag

```
uint16 CFE_TBL_ValidateCmd_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 validated

Definition at line 560 of file cfe_tbl_msg.h.

11.119.2.2 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 566 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h

11.120 CFE_TIME_DiagnosticTlm Struct Reference

```
#include <cfе_time_msg.h>
```

Data Fields

- [CFE_MSG_TelemetryHeader_t TelemetryHeader](#)
Telemetry header.
- [CFE_TIME_DiagnosticTlm_Payload_t Payload](#)
Telemetry payload.

11.120.1 Detailed Description

Definition at line 1122 of file cfe_time_msg.h.

11.120.2 Field Documentation

11.120.2.1 Payload

[CFE_TIME_DiagnosticTlm_Payload_t](#) CFE_TIME_DiagnosticTlm::Payload

Telemetry payload.

Definition at line 1125 of file cfe_time_msg.h.

11.120.2.2 TelemetryHeader

[CFE_MSG_TelemetryHeader_t](#) CFE_TIME_DiagnosticTlm::TelemetryHeader

Telemetry header.

Definition at line 1124 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/[cfе_time_msg.h](#)

11.121 CFE_TIME_DiagnosticTlm_Payload Struct Reference

```
#include <cfе_time_msg.h>
```

Data Fields

- [CFE_TIME_SysTime_t AtToneMET](#)
MET at time of tone.
- [CFE_TIME_SysTime_t AtToneSTCF](#)
STCF at time of tone.
- [CFE_TIME_SysTime_t AtToneDelay](#)
Adjustment for slow tone detection.
- [CFE_TIME_SysTime_t AtToneLatch](#)
Local clock latched at time of tone.
- [int16 AtToneLeapSeconds](#)
Leap Seconds at time of tone.
- [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](#)

- **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)

11.121.1 Detailed Description

Name Time Services Diagnostics Packet

Definition at line 976 of file cfe_time_msg.h.

11.121.2 Field Documentation

11.121.2.1 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 985 of file `cfe_time_msg.h`.

11.121.2.2 AtToneLatch

`CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::AtToneLatch`

Local clock latched at time of tone.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DTVvalidS`, `$sc_$cpu_TIME_DTVvalidSs`

Definition at line 987 of file `cfe_time_msg.h`.

11.121.2.3 AtToneLeapSeconds

`int16` `CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds`

Leap Seconds at time of tone.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DLapS`

Definition at line 990 of file `cfe_time_msg.h`.

11.121.2.4 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_DTMETSSs`

Definition at line 981 of file `cfe_time_msg.h`.

11.121.2.5 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 983 of file `cfe_time_msg.h`.

11.121.2.6 ClockFlyState

`int16` `CFE_TIME_DiagnosticTlm_Payload::ClockFlyState`

Current fly-wheel state.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DFlywheel`

Definition at line 1014 of file `cfe_time_msg.h`.

11.121.2.7 ClockSetState

`int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSetState`

Time has been "set".

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DValid`

Definition at line 1012 of file `cfe_time_msg.h`.

11.121.2.8 ClockSignal

`int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSignal`

Primary vs redundant, etc.

Telemetry Mnemonic(s) `$sc_$cpu_TIME_DSignal`

Definition at line 1018 of file `cfe_time_msg.h`.

11.121.2.9 ClockSource

`int16 CFE_TIME_DiagnosticTlm_Payload::ClockSource`

Internal vs external, etc.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DSource

Definition at line 1016 of file cfe_time_msg.h.

11.121.2.10 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 992 of file cfe_time_msg.h.

11.121.2.11 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 1028 of file cfe_time_msg.h.

11.121.2.12 CurrentLatch

`CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload::CurrentLatch`

Local clock latched just "now".

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLocalS, \$sc_\$cpu_TIME_DLocalSs

Definition at line 1000 of file cfe_time_msg.h.

11.121.2.13 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 1002 of file `cfe_time_msg.h`.

11.121.2.14 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 1004 of file `cfe_time_msg.h`.

11.121.2.15 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 1006 of file `cfe_time_msg.h`.

11.121.2.16 DataStoreStatus

`uint32` `CFE_TIME_DiagnosticTlm_Payload::DataStoreStatus`

Data Store status (preserved across processor reset)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DataStStat

Definition at line 1118 of file `cfe_time_msg.h`.

11.121.2.17 DelayDirection

```
int16 CFE_TIME_DiagnosticTlm_Payload::DelayDirection
```

Client latency adjustment direction.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLlatentDir

Definition at line 1038 of file cfe_time_msg.h.

11.121.2.18 Forced2Fly

```
int16 CFE_TIME_DiagnosticTlm_Payload::Forced2Fly
```

Commanded into fly-wheel.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DCMD2Fly

Definition at line 1022 of file cfe_time_msg.h.

11.121.2.19 LocalIntCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::LocalIntCounter
```

Local 1Hz ISR execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzISRCNT

Definition at line 1076 of file cfe_time_msg.h.

11.121.2.20 LocalTaskCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::LocalTaskCounter
```

Local 1Hz task execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzTaskCNT

Definition at line 1078 of file cfe_time_msg.h.

11.121.2.21 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 1098 of file cfe_time_msg.h.

11.121.2.22 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 1104 of file cfe_time_msg.h.

11.121.2.23 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 1096 of file cfe_time_msg.h.

11.121.2.24 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 1046 of file cfe_time_msg.h.

11.121.2.25 OneHzDirection

`int16 CFE_TIME_DiagnosticTlm_Payload::OneHzDirection`

1Hz STCF adjustment direction

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_D1HzAdjDir

Definition at line 1036 of file cfe_time_msg.h.

11.121.2.26 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 1044 of file cfe_time_msg.h.

11.121.2.27 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 1034 of file cfe_time_msg.h.

11.121.2.28 ServerFlyState

`int16 CFE_TIME_DiagnosticTlm_Payload::ServerFlyState`

Used by clients only.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DSrvFly

Definition at line 1020 of file cfe_time_msg.h.

11.121.2.29 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 998 of file cfe_time_msg.h.

11.121.2.30 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 1066 of file cfe_time_msg.h.

11.121.2.31 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 1054 of file cfe_time_msg.h.

11.121.2.32 ToneIntCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneIntCounter
```

Tone signal ISR execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTsISRCNT

Definition at line 1068 of file cfe_time_msg.h.

11.121.2.33 ToneIntErrorCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneIntErrorCounter
```

Tone signal ISR error count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTslSRERR

Definition at line 1070 of file cfe_time_msg.h.

11.121.2.34 ToneMatchCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneMatchCounter
```

Tone signal / data verification count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DVerifyCNT

Definition at line 1060 of file cfe_time_msg.h.

11.121.2.35 ToneMatchErrorCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneMatchErrorCounter
```

Tone signal / data verification error count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DVerifyER

Definition at line 1062 of file cfe_time_msg.h.

11.121.2.36 ToneOverLimit

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneOverLimit
```

Max between tone signal interrupts.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DMaxSs

Definition at line 1110 of file cfe_time_msg.h.

11.121.2.37 ToneSignalCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneSignalCounter
```

Tone signal detected SB message count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTSDetCNT

Definition at line 1064 of file cfe_time_msg.h.

11.121.2.38 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 1052 of file cfe_time_msg.h.

11.121.2.39 ToneTaskCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneTaskCounter
```

Tone task execution count.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DTsTaskCNT

Definition at line 1072 of file cfe_time_msg.h.

11.121.2.40 ToneUnderLimit

```
uint32 CFE_TIME_DiagnosticTlm_Payload::ToneUnderLimit
```

Min between tone signal interrupts.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DMinSs

Definition at line 1112 of file cfe_time_msg.h.

11.121.2.41 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 1074 of file cfe_time_msg.h.

11.121.2.42 VirtualMET

`uint32 CFE_TIME_DiagnosticTlm_Payload::VirtualMET`

Software MET.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DLLogicalMET

Definition at line 1084 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/cfe_time_msg.h

11.122 CFE_TIME_HousekeepingTlm Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`
Telemetry header.
- `CFE_TIME_HousekeepingTlm_Payload_t Payload`
Telemetry payload.

11.122.1 Detailed Description

Definition at line 965 of file cfe_time_msg.h.

11.122.2 Field Documentation

11.122.2.1 Payload

`CFE_TIME_HousekeepingTlm_Payload_t` CFE_TIME_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 968 of file `cfe_time_msg.h`.

11.122.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t` CFE_TIME_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 967 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.123 CFE_TIME_HousekeepingTlm_Payload Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- `uint8 CommandCounter`
Time Command Execution Counter.
- `uint8 CommandErrorCounter`
Time Command Error Counter.
- `uint16 ClockStateFlags`
State Flags.
- `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)*

11.123.1 Detailed Description

Name Time Services Housekeeping Packet

Definition at line 906 of file cfe_time_msg.h.

11.123.2 Field Documentation

11.123.2.1 ClockStateAPI

`CFE_TIME_ClockState_Enum_t` `CFE_TIME_HousekeepingTlm_Payload::ClockStateAPI`

API State.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_DAPIStrate

Definition at line 921 of file cfe_time_msg.h.

11.123.2.2 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_FlagAdj, \$sc_\$cpu_TIME_Flag1Hzd, \$sc_\$cpu_TIME_FlagClat, \$sc_\$cpu_TIME_FlagSorC, \$sc_\$cpu_TIME_FlagNIU

Definition at line 919 of file cfe_time_msg.h.

11.123.2.3 CommandCounter

```
uint8 CFE_TIME_HousekeepingTlm_Payload::CommandCounter
```

Time Command Execution Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_CMDPC

Definition at line 911 of file cfe_time_msg.h.

11.123.2.4 CommandErrorCounter

```
uint8 CFE_TIME_HousekeepingTlm_Payload::CommandErrorCounter
```

Time Command Error Counter.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_CMDEC

Definition at line 913 of file cfe_time_msg.h.

11.123.2.5 LeapSeconds

```
int16 CFE_TIME_HousekeepingTlm_Payload::LeapSeconds
```

Current Leaps Seconds.

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_LeapSecs

Definition at line 927 of file cfe_time_msg.h.

11.123.2.6 Seconds1HzAdj

```
uint32 CFE_TIME_HousekeepingTlm_Payload::Seconds1HzAdj
```

Current 1 Hz SCTF adjustment (seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_1HzAdjSecs

Definition at line 947 of file cfe_time_msg.h.

11.123.2.7 SecondsDelay

`uint32 CFE_TIME_HousekeepingTlm_Payload::SecondsDelay`

Current 1 Hz SCTF Delay (seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_1HzAdjSecs

Definition at line 957 of file cfe_time_msg.h.

11.123.2.8 SecondsMET

`uint32 CFE_TIME_HousekeepingTlm_Payload::SecondsMET`

Current MET (seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_METSecs

Definition at line 933 of file cfe_time_msg.h.

11.123.2.9 SecondsSTCF

`uint32 CFE_TIME_HousekeepingTlm_Payload::SecondsSTCF`

Current STCF (seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_STCFSecs

Definition at line 938 of file cfe_time_msg.h.

11.123.2.10 Subsecs1HzAdj

`uint32 CFE_TIME_HousekeepingTlm_Payload::Subsecs1HzAdj`

Current 1 Hz SCTF adjustment (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_1HzAdjSSecs

Definition at line 949 of file cfe_time_msg.h.

11.123.2.11 SubsecsDelay

`uint32 CFE_TIME_HousekeepingTlm_Payload::SubsecsDelay`

Current 1 Hz SCTF Delay (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_1HzAdjSSecs

Definition at line 959 of file cfe_time_msg.h.

11.123.2.12 SubsecsMET

`uint32 CFE_TIME_HousekeepingTlm_Payload::SubsecsMET`

Current MET (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_METSubsecs

Definition at line 935 of file cfe_time_msg.h.

11.123.2.13 SubsecsSTCF

`uint32 CFE_TIME_HousekeepingTlm_Payload::SubsecsSTCF`

Current STCF (sub-seconds)

Telemetry Mnemonic(s) \$sc_\$cpu_TIME_STCFSubsecs

Definition at line 940 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/cfe_time_msg.h

11.124 CFE_TIME_LeapsCmd_Payload Struct Reference

Set leap seconds command payload.

```
#include <cfe_time_msg.h>
```

Data Fields

- `int16 LeapSeconds`

11.124.1 Detailed Description

Set leap seconds command payload.

Definition at line 750 of file `cfe_time_msg.h`.

11.124.2 Field Documentation**11.124.2.1 LeapSeconds**

`int16 CFE_TIME_LeapsCmd_Payload::LeapSeconds`

Definition at line 752 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.125 CFE_TIME_NoArgsCmd Struct Reference

Generic no argument command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.

11.125.1 Detailed Description

Generic no argument command.

Definition at line 729 of file `cfe_time_msg.h`.

11.125.2 Field Documentation

11.125.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_TIME_NoArgsCmd::CommandHeader`

Command header.

Definition at line 731 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.126 CFE_TIME_OneHzAdjustmentCmd Struct Reference

Generic seconds, subseconds adjustment command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_OneHzAdjustmentCmd_Payload_t Payload`
Command payload.

11.126.1 Detailed Description

Generic seconds, subseconds adjustment command.

Definition at line 867 of file `cfe_time_msg.h`.

11.126.2 Field Documentation

11.126.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_TIME_OneHzAdjustmentCmd::CommandHeader`

Command header.

Definition at line 869 of file `cfe_time_msg.h`.

11.126.2.2 Payload

`CFE_TIME_OneHzAdjustmentCmd_Payload_t CFE_TIME_OneHzAdjustmentCmd::Payload`

Command payload.

Definition at line 870 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.127 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference

Generic seconds, subseconds command payload.

```
#include <cfe_time_msg.h>
```

Data Fields

- `uint32 Seconds`
- `uint32 Subseconds`

11.127.1 Detailed Description

Generic seconds, subseconds command payload.

Definition at line 857 of file `cfe_time_msg.h`.

11.127.2 Field Documentation

11.127.2.1 Seconds

`uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Seconds`

Definition at line 859 of file `cfe_time_msg.h`.

11.127.2.2 Subseconds

`uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Subseconds`

Definition at line 860 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.128 CFE_TIME_SetLeapSecondsCmd Struct Reference

Set leap seconds command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_LeapsCmd_Payload_t Payload`
Command payload.

11.128.1 Detailed Description

Set leap seconds command.

Definition at line 758 of file `cfe_time_msg.h`.

11.128.2 Field Documentation

11.128.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_TIME_SetLeapSecondsCmd::CommandHeader`

Command header.

Definition at line 760 of file `cfe_time_msg.h`.

11.128.2.2 Payload

`CFE_TIME_LeapsCmd_Payload_t CFE_TIME_SetLeapSecondsCmd::Payload`

Command payload.

Definition at line 761 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.129 CFE_TIME_SetSignalCmd Struct Reference

Set tone signal source command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_SignalCmd_Payload_t Payload`
Command payload.

11.129.1 Detailed Description

Set tone signal source command.

Definition at line 817 of file `cfe_time_msg.h`.

11.129.2 Field Documentation

11.129.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_TIME_SetSignalCmd::CommandHeader`

Command header.

Definition at line 819 of file `cfe_time_msg.h`.

11.129.2.2 Payload

`CFE_TIME_SignalCmd_Payload_t CFE_TIME_SetSignalCmd::Payload`

Command payload.

Definition at line 820 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.130 CFE_TIME_SetSourceCmd Struct Reference

Set time data source command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_SourceCmd_Payload_t Payload`
Command payload.

11.130.1 Detailed Description

Set time data source command.

Definition at line 798 of file `cfe_time_msg.h`.

11.130.2 Field Documentation

11.130.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_TIME_SetSourceCmd::CommandHeader`

Command header.

Definition at line 800 of file `cfe_time_msg.h`.

11.130.2.2 Payload

`CFE_TIME_SourceCmd_Payload_t CFE_TIME_SetSourceCmd::Payload`

Command payload.

Definition at line 801 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.131 CFE_TIME_SetStateCmd Struct Reference

Set clock state command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_StateCmd_Payload_t Payload`
Command payload.

11.131.1 Detailed Description

Set clock state command.

Definition at line 779 of file `cfe_time_msg.h`.

11.131.2 Field Documentation

11.131.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_TIME_SetStateCmd::CommandHeader`

Command header.

Definition at line 781 of file `cfe_time_msg.h`.

11.131.2.2 Payload

`CFE_TIME_StateCmd_Payload_t` `CFE_TIME_SetStateCmd::Payload`

Command payload.

Definition at line 782 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.132 CFE_TIME_SignalCmd_Payload Struct Reference

Set tone signal source command payload.

```
#include <cfe_time_msg.h>
```

Data Fields

- `int16 ToneSource`
`CFE_TIME_ToneSignalSelect_PRIMARY`=Primary Source, `CFE_TIME_ToneSignalSelect_REDUNDANT`=Redundant Source

11.132.1 Detailed Description

Set tone signal source command payload.

Definition at line 807 of file `cfe_time_msg.h`.

11.132.2 Field Documentation

11.132.2.1 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 809 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.133 CFE_TIME_SourceCmd_Payload Struct Reference

Set time data source command payload.

```
#include <cfе_time_msg.h>
```

Data Fields

- `int16 TimeSource`
`CFE_TIME_SourceSelect_INTERNAL`=Internal Source, `CFE_TIME_SourceSelect_EXTERNAL`=External Source

11.133.1 Detailed Description

Set time data source command payload.

Definition at line 788 of file cfe_time_msg.h.

11.133.2 Field Documentation

11.133.2.1 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 790 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.134 CFE_TIME_StateCmd_Payload Struct Reference

Set clock state command payload.

```
#include <cfе_time_msg.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

11.134.1 Detailed Description

Set clock state command payload.

Definition at line 767 of file cfe_time_msg.h.

11.134.2 Field Documentation

11.134.2.1 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 769 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/cfe_time_msg.h

11.135 CFE_TIME_SysTime Struct Reference

Data structure used to hold system time values.

```
#include <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)

11.135.1 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 51 of file cfe_time_extern_typedefs.h.

11.135.2 Field Documentation

11.135.2.1 Seconds

`uint32 CFE_TIME_SysTime::Seconds`

Number of seconds since epoch.

Definition at line 53 of file `cfe_time_extern_typedefs.h`.

11.135.2.2 Subseconds

`uint32 CFE_TIME_SysTime::Subseconds`

Number of subseconds since epoch (LSB = 2^{-32} seconds)

Definition at line 54 of file `cfe_time_extern_typedefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h`

11.136 CFE_TIME_TimeCmd Struct Reference

Generic seconds, microseconds argument command.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`
Command header.
- `CFE_TIME_TimeCmd_Payload_t Payload`
Command payload.

11.136.1 Detailed Description

Generic seconds, microseconds argument command.

Definition at line 835 of file `cfe_time_msg.h`.

11.136.2 Field Documentation

11.136.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_TIME_TimeCmd::CommandHeader`

Command header.

Definition at line 837 of file `cfe_time_msg.h`.

11.136.2.2 Payload

`CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_TimeCmd::Payload`

Command payload.

Definition at line 838 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.137 CFE_TIME_TimeCmd_Payload Struct Reference

Generic seconds, microseconds command payload.

```
#include <cfe_time_msg.h>
```

Data Fields

- `uint32 Seconds`
- `uint32 MicroSeconds`

11.137.1 Detailed Description

Generic seconds, microseconds command payload.

Definition at line 826 of file `cfe_time_msg.h`.

11.137.2 Field Documentation

11.137.2.1 MicroSeconds

```
uint32 CFE_TIME_TimeCmd_Payload::MicroSeconds
```

Definition at line 829 of file cfe_time_msg.h.

11.137.2.2 Seconds

```
uint32 CFE_TIME_TimeCmd_Payload::Seconds
```

Definition at line 828 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/cfe_time_msg.h

11.138 CFE_TIME_ToneDataCmd Struct Reference

Time at tone data command.

```
#include <cfe_time_msg.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CommandHeader](#)
Command header.
- [CFE_TIME_ToneDataCmd_Payload_t Payload](#)
Command payload.

11.138.1 Detailed Description

Time at tone data command.

Definition at line 895 of file cfe_time_msg.h.

11.138.2 Field Documentation

11.138.2.1 CommandHeader

`CFE_MSG_CommandHeader_t` `CFE_TIME_ToneDataCmd::CommandHeader`

Command header.

Definition at line 897 of file `cfe_time_msg.h`.

11.138.2.2 Payload

`CFE_TIME_ToneDataCmd_Payload_t` `CFE_TIME_ToneDataCmd::Payload`

Command payload.

Definition at line 898 of file `cfe_time_msg.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/fsw/inc/cfe_time_msg.h`

11.139 CFE_TIME_ToneDataCmd_Payload Struct Reference

Time at tone data command payload.

```
#include <cfe_time_msg.h>
```

Data Fields

- `CFE_TIME_SysTime_t AtToneMET`
MET at time of tone.
- `CFE_TIME_SysTime_t AtToneSTCF`
STCF at time of tone.
- `int16 AtToneLeapSeconds`
Leap Seconds at time of tone.
- `CFE_TIME_ClockState_Enum_t AtToneState`
Clock state at time of tone.

11.139.1 Detailed Description

Time at tone data command payload.

Definition at line 884 of file `cfe_time_msg.h`.

11.139.2 Field Documentation

11.139.2.1 AtToneLeapSeconds

```
int16 CFE_TIME_ToneDataCmd_Payload::AtToneLeapSeconds
```

Leap Seconds at time of tone.

Definition at line 888 of file cfe_time_msg.h.

11.139.2.2 AtToneMET

```
CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneMET
```

MET at time of tone.

Definition at line 886 of file cfe_time_msg.h.

11.139.2.3 AtToneState

```
CFE_TIME_ClockState_Enum_t CFE_TIME_ToneDataCmd_Payload::AtToneState
```

Clock state at time of tone.

Definition at line 889 of file cfe_time_msg.h.

11.139.2.4 AtToneSTCF

```
CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneSTCF
```

STCF at time of tone.

Definition at line 887 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/fsw/inc/[cfe_time_msg.h](#)

11.140 HS_AMTEntry_t Struct Reference

Application Monitor Table (AMT) Entry.

```
#include <hs_tbl.h>
```

Data Fields

- `char AppName [OS_MAX_API_NAME]`
Name of application to be monitored.
- `uint16 NullTerm`
Buffer of nulls to terminate string.
- `uint16 CycleCount`
Number of cycles before application is missing.
- `uint16 ActionType`
Action to take if application is missing.

11.140.1 Detailed Description

Application Monitor Table (AMT) Entry.

Definition at line 66 of file hs_tbl.h.

11.140.2 Field Documentation

11.140.2.1 ActionType

```
uint16 HS_AMTEntry_t::ActionType
```

Action to take if application is missing.

Definition at line 71 of file hs_tbl.h.

Referenced by HS_AppMonStatusRefresh(), HS_MonitorApplications(), and HS_ValidateAMTable().

11.140.2.2 AppName

```
char HS_AMTEntry_t::AppName [OS_MAX_API_NAME]
```

Name of application to be monitored.

Definition at line 68 of file hs_tbl.h.

Referenced by HS_MonitorApplications().

11.140.2.3 CycleCount

```
uint16 HS_AMTEntry_t::CycleCount
```

Number of cycles before application is missing.

Definition at line 70 of file hs_tbl.h.

Referenced by HS_AppMonStatusRefresh(), HS_MonitorApplications(), and HS_ValidateAMTable().

11.140.2.4 NullTerm

```
uint16 HS_AMTEntry_t::NullTerm
```

Buffer of nulls to terminate string.

Definition at line 69 of file hs_tbl.h.

Referenced by HS_ValidateAMTable().

The documentation for this struct was generated from the following file:

- [apps/hs/fsw/src/hs_tbl.h](#)

11.141 HS_AppData_t Struct Reference

HS Global Data Structure.

```
#include <hs_app.h>
```

Data Fields

- [CFE_SB_Pipeld_t CmdPipe](#)
Pipe Id for HS command pipe.
- [CFE_SB_Pipeld_t WakeupPipe](#)
Pipe Id for HS wakeup pipe.
- [CFE_SB_Pipeld_t EventPipe](#)
Pipe Id for HK event pipe.
- [uint8 ServiceWatchdogFlag](#)
Flag of current watchdog servicing state.
- [uint8 CurrentAppMonState](#)
Status of HS Application Monitor.
- [uint8 CurrentEventMonState](#)
Status of HS Events Monitor.
- [uint8 CurrentAlivenessState](#)
Status of HS Aliveness Indicator.

- `uint8 ExeCountState`
Status of Execution Counter Table.
- `uint8 MsgActsState`
Status of Message Actions Table.
- `uint8 CDSState`
Status of Critical Data Storing.
- `uint8 AppMonLoaded`
If AppMon Table is loaded.
- `uint8 EventMonLoaded`
If EventMon Table is loaded.
- `uint8 CurrentCPUHogState`
Status of HS CPU Hogging Indicator.
- `uint8 SpareBytes [3]`
Spare bytes for 32 bit alignment padding.
- `uint8 CmdCount`
Number of valid commands received.
- `uint8 CmdErrCount`
Number of invalid commands received.
- `uint32 EventsMonitoredCount`
Total count of event messages monitored.
- `uint16 MsgActCooldown [HS_MAX_MSG_ACT_TYPES]`
Counts until Message Actions is available.
- `uint16 AppMonCheckInCountdown [HS_MAX_MONITORED_APPS]`
Counts until Application Monitor times out.
- `uint32 AppMonEnables [(HS_MAX_MONITORED_APPS - 1)/HS_BITS_PER_APPMON_ENABLE)+1]`
AppMon state by monitor.
- `uint32 AppMonLastExeCount [HS_MAX_MONITORED_APPS]`
Last Execution Count for application being checked.
- `uint32 AlivenessCounter`
Current Count towards the CPU Aliveness output period.
- `uint32 MsgActExec`
Number of Software Bus Message Actions Executed.
- `uint32 RunStatus`
HS App run status.
- `uint32 CurrentCPUHoggingTime`
Count of cycles that CPU utilization is above hogging threshold.
- `uint32 MaxCPUHoggingTime`
Count of hogging cycles after which an event reports hogging.
- `uint32 CurrentCPUUtilIndex`
Current index into the Utilization Tracker.
- `uint32 UtilizationTracker [HS_UTIL_PEAK_NUM_INTERVAL]`
Utilization from previous intervals.
- `uint32 UtilCpuAvg`
Current CPU Utilization Average.
- `uint32 UtilCpuPeak`
Current CPU Utilization Peak.
- `CFE_TBL_Handle_t AMTableHandle`

- *Apps Monitor table handle.*
 - **CFE_TBL_Handle_t** **EMTableHandle**
Events Monitor table handle.
 - **CFE_TBL_Handle_t** **MATableHandle**
Message Actions table handle.
- **HS_AMTEntry_t *** **AMTablePtr**
Ptr to Apps Monitor table entry.
- **HS_EMTEntry_t *** **EMTablePtr**
Ptr to Events Monitor table entry.
- **HS_MATEEntry_t *** **MATablePtr**
Ptr to Message Actions table entry.
- **CFE_ES_CDSHandle_t** **MyCDSHandle**
- **HS_CDSDData_t** **CDSDData**
- **HS_HkPacket_t** **HkPacket**
HK Housekeeping Packet.

11.141.1 Detailed Description

HS Global Data Structure.

Definition at line 77 of file hs_app.h.

11.141.2 Field Documentation

11.141.2.1 AlivenessCounter

```
uint32 HS_AppData_t::AlivenessCounter
```

Current Count towards the CPU Aliveness output period.

Definition at line 111 of file hs_app.h.

Referenced by HS_ProcessMain().

11.141.2.2 AMTableHandle

```
CFE_TBL_Handle_t HS_AppData_t::AMTableHandle
```

Apps Monitor table handle.

Definition at line 126 of file hs_app.h.

Referenced by HS_AcquirePointers(), and HS_TblInit().

11.141.2.3 AMTablePtr

```
HS_AMTEntry_t* HS_AppData_t::AMTablePtr
```

Ptr to Apps Monitor table entry.

Definition at line 135 of file hs_app.h.

Referenced by HS_AcquirePointers(), HS_AppMonStatusRefresh(), and HS_MonitorApplications().

11.141.2.4 AppMonCheckInCountdown

```
uint16 HS_AppData_t::AppMonCheckInCountdown[HS_MAX_MONITORED_APPS]
```

Counts until Application Monitor times out.

Definition at line 104 of file hs_app.h.

Referenced by HS_AppMonStatusRefresh(), and HS_MonitorApplications().

11.141.2.5 AppMonEnables

```
uint32 HS_AppData_t::AppMonEnables[((HS_MAX_MONITORED_APPS - 1) / HS_BITS_PER_APPMON_ENABLE) + 1]
```

AppMon state by monitor.

Definition at line 107 of file hs_app.h.

Referenced by HS_AppMonStatusRefresh(), HS_HousekeepingReq(), and HS_MonitorApplications().

11.141.2.6 AppMonLastExeCount

```
uint32 HS_AppData_t::AppMonLastExeCount[HS_MAX_MONITORED_APPS]
```

Last Execution Count for application being checked.

Definition at line 109 of file hs_app.h.

Referenced by HS_AppMonStatusRefresh(), and HS_MonitorApplications().

11.141.2.7 AppMonLoaded

```
uint8 HS_AppData_t::AppMonLoaded
```

If AppMon Table is loaded.

Definition at line 92 of file hs_app.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_HousekeepingReq(), and HS_TblInit().

11.141.2.8 CDSData

```
HS_CDSDData_t HS_AppData_t::CDSData
```

Definition at line 140 of file hs_app.h.

Referenced by HS_AppInit(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_Reset← ResetsPerformedCmd(), HS_SetCDSDData(), and HS_SetMaxResetsCmd().

11.141.2.9 CDSState

```
uint8 HS_AppData_t::CDSState
```

Status of Critical Data Storing.

Definition at line 91 of file hs_app.h.

Referenced by HS_AppInit(), HS_HousekeepingReq(), and HS_SetCDSDData().

11.141.2.10 CmdCount

```
uint8 HS_AppData_t::CmdCount
```

Number of valid commands received.

Definition at line 98 of file hs_app.h.

Referenced by HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_Disable← EventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_Enable← EventMonCmd(), HS_HousekeepingReq(), HS_NoopCmd(), HS_ResetCounters(), HS_ResetResetsPerformedCmd(), HS_SetMaxResetsCmd(), HS_SetUtilDiagCmd(), and HS_SetUtilParamsCmd().

11.141.2.11 CmdErrCount

```
uint8 HS_AppData_t::CmdErrCount
```

Number of invalid commands received.

Definition at line 99 of file hs_app.h.

Referenced by HS_AppPipe(), HS_DisableEventMonCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_ResetCounters(), HS_SetUtilParamsCmd(), and HS_VerifyMsgLength().

11.141.2.12 CmdPipe

```
CFE_SB_PipeId_t HS_AppData_t::CmdPipe
```

Pipe Id for HS command pipe.

Definition at line 79 of file hs_app.h.

Referenced by HS_ProcessCommands(), and HS_SbInit().

11.141.2.13 CurrentAlivenessState

```
uint8 HS_AppData_t::CurrentAlivenessState
```

Status of HS Aliveness Indicator.

Definition at line 87 of file hs_app.h.

Referenced by HS_AppInit(), HS_DisableAlivenessCmd(), HS_EnableAlivenessCmd(), HS_HousekeepingReq(), and HS_ProcessMain().

11.141.2.14 CurrentAppMonState

```
uint8 HS_AppData_t::CurrentAppMonState
```

Status of HS Application Monitor.

Definition at line 85 of file hs_app.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_DisableAppMonCmd(), HS_EnableAppMonCmd(), HS_HousekeepingReq(), HS_ProcessMain(), and HS_TblInit().

11.141.2.15 CurrentCPUHoggingTime

```
uint32 HS_AppData_t::CurrentCPUHoggingTime
```

Count of cycles that CPU utilization is above hogging threshold.

Definition at line 117 of file hs_app.h.

Referenced by HS_MonitorUtilization().

11.141.2.16 CurrentCPUHogState

```
uint8 HS_AppData_t::CurrentCPUHogState
```

Status of HS CPU Hogging Indicator.

Definition at line 95 of file hs_app.h.

Referenced by HS_AppInit(), HS_DisableCPUHogCmd(), HS_EnableCPUHogCmd(), HS_HousekeepingReq(), and HS_MonitorUtilization().

11.141.2.17 CurrentCPUUtilIndex

```
uint32 HS_AppData_t::CurrentCPUUtilIndex
```

Current index into the Utilization Tracker.

Definition at line 119 of file hs_app.h.

Referenced by HS_MonitorUtilization().

11.141.2.18 CurrentEventMonState

```
uint8 HS_AppData_t::CurrentEventMonState
```

Status of HS Events Monitor.

Definition at line 86 of file hs_app.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_AppMain(), HS_DisableEventMonCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_ProcessCommands(), and HS_TblInit().

11.141.2.19 EMTTableHandle

`CFE_TBL_Handle_t HS_AppData_t::EMTTableHandle`

Events Monitor table handle.

Definition at line 127 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, and `HS_TblInit()`.

11.141.2.20 EMTTablePtr

`HS_EMTEntry_t* HS_AppData_t::EMTTablePtr`

Ptr to Events Monitor table entry.

Definition at line 136 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, `HS_HousekeepingReq()`, and `HS_MonitorEvent()`.

11.141.2.21 EventMonLoaded

`uint8 HS_AppData_t::EventMonLoaded`

If EventMon Table is loaded.

Definition at line 93 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, `HS_AppInit()`, `HS_HousekeepingReq()`, and `HS_TblInit()`.

11.141.2.22 EventPipe

`CFE_SB_PipeId_t HS_AppData_t::EventPipe`

Pipe Id for HK event pipe.

Definition at line 81 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, `HS_AppMain()`, `HS_DisableEventMonCmd()`, `HS_EnableEventMonCmd()`, `HS_ProcessCommands()`, and `HS_SblInit()`.

11.141.2.23 EventsMonitoredCount

`uint32 HS_AppData_t::EventsMonitoredCount`

Total count of event messages monitored.

Definition at line 101 of file `hs_app.h`.

Referenced by `HS_HousekeepingReq()`, `HS_ProcessCommands()`, and `HS_ResetCounters()`.

11.141.2.24 ExeCountState

`uint8 HS_AppData_t::ExeCountState`

Status of Execution Counter Table.

Definition at line 88 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, `HS_AppInit()`, `HS_HousekeepingReq()`, and `HS_TblInit()`.

11.141.2.25 HkPacket

`HS_HkPacket_t HS_AppData_t::HkPacket`

HK Housekeeping Packet.

Definition at line 142 of file `hs_app.h`.

Referenced by `HS_HousekeepingReq()`, `HS_SblInit()`, and `HS_TblInit()`.

11.141.2.26 MATableHandle

`CFE_TBL_Handle_t HS_AppData_t::MATableHandle`

Message Actions table handle.

Definition at line 128 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, and `HS_TblInit()`.

11.141.2.27 MATablePtr

```
HS_MATEntry_t* HS_AppData_t::MATablePtr
```

Ptr to Message Actions table entry.

Definition at line 137 of file hs_app.h.

Referenced by HS_AcquirePointers(), HS_MonitorApplications(), and HS_MonitorEvent().

11.141.2.28 MaxCPUHoggingTime

```
uint32 HS_AppData_t::MaxCPUHoggingTime
```

Count of hogging cycles after which an event reports hogging.

Definition at line 118 of file hs_app.h.

Referenced by HS_AppInit(), and HS_MonitorUtilization().

11.141.2.29 MsgActCooldown

```
uint16 HS_AppData_t::MsgActCooldown[HS_MAX_MSG_ACT_TYPES]
```

Counts until Message Actions is available.

Definition at line 103 of file hs_app.h.

Referenced by HS_MonitorApplications(), HS_MonitorEvent(), HS_MsgActsStatusRefresh(), and HS_ProcessMain().

11.141.2.30 MsgActExec

```
uint32 HS_AppData_t::MsgActExec
```

Number of Software Bus Message Actions Executed.

Definition at line 113 of file hs_app.h.

Referenced by HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), and HS_ResetCounters().

11.141.2.31 MsgActsState

`uint8 HS_AppData_t::MsgActsState`

Status of Message Actions Table.

Definition at line 90 of file `hs_app.h`.

Referenced by `HS_AcquirePointers()`, `HS_AppInit()`, `HS_HousekeepingReq()`, `HS_MonitorApplications()`, `HS_MonitorEvent()`, and `HS_TblInit()`.

11.141.2.32 MyCDSHandle

`CFE_ES_CDSHandle_t HS_AppData_t::MyCDSHandle`

Definition at line 139 of file `hs_app.h`.

Referenced by `HS_AppInit()`, and `HS_SetCDSData()`.

11.141.2.33 RunStatus

`uint32 HS_AppData_t::RunStatus`

HS App run status.

Definition at line 115 of file `hs_app.h`.

Referenced by `HS_AppInit()`.

11.141.2.34 ServiceWatchdogFlag

`uint8 HS_AppData_t::ServiceWatchdogFlag`

Flag of current watchdog servicing state.

Definition at line 83 of file `hs_app.h`.

Referenced by `HS_AppInit()`, `HS_MonitorApplications()`, `HS_MonitorEvent()`, and `HS_ProcessMain()`.

11.141.2.35 SpareBytes

```
uint8 HS_AppData_t::SpareBytes[3]
```

Spare bytes for 32 bit alignment padding.

Definition at line 96 of file hs_app.h.

11.141.2.36 UtilCpuAvg

```
uint32 HS_AppData_t::UtilCpuAvg
```

Current CPU Utilization Average.

Definition at line 123 of file hs_app.h.

Referenced by HS_HousekeepingReq(), and HS_MonitorUtilization().

11.141.2.37 UtilCpuPeak

```
uint32 HS_AppData_t::UtilCpuPeak
```

Current CPU Utilization Peak.

Definition at line 124 of file hs_app.h.

Referenced by HS_HousekeepingReq(), and HS_MonitorUtilization().

11.141.2.38 UtilizationTracker

```
uint32 HS_AppData_t::UtilizationTracker[HS_UTIL_PEAK_NUM_INTERVAL]
```

Utilization from previous intervals.

Definition at line 121 of file hs_app.h.

Referenced by HS_MonitorUtilization().

11.141.2.39 WakeupPipe

`CFE_SB_PipeId_t HS_AppData_t::WakeupPipe`

Pipe Id for HS wakeup pipe.

Definition at line 80 of file `hs_app.h`.

Referenced by `HS_AppMain()`, and `HS_SbInit()`.

The documentation for this struct was generated from the following file:

- `apps/hs/fsw/src/hs_app.h`

11.142 HS_CDSData_t Struct Reference

HS CDS Data Structure.

```
#include <hs_app.h>
```

Data Fields

- **`uint16 ResetsPerformed`**
Number of Resets Performed.
- **`uint16 ResetsPerformedNot`**
Inverted Resets Performed for validation.
- **`uint16 MaxResets`**
Max Number of Resets Allowed.
- **`uint16 MaxResetsNot`**
Inverted Max Number of Resets Allowed for validation.

11.142.1 Detailed Description

HS CDS Data Structure.

Definition at line 66 of file `hs_app.h`.

11.142.2 Field Documentation

11.142.2.1 MaxResets

```
uint16 HS_CDSData_t::MaxResets
```

Max Number of Resets Allowed.

Definition at line 70 of file hs_app.h.

Referenced by HS_AppInit(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_Reset← ResetsPerformedCmd(), HS_SetCDSData(), and HS_SetMaxResetsCmd().

11.142.2.2 MaxResetsNot

```
uint16 HS_CDSData_t::MaxResetsNot
```

Inverted Max Number of Resets Allowed for validation.

Definition at line 71 of file hs_app.h.

Referenced by HS_AppInit(), and HS_SetCDSData().

11.142.2.3 ResetsPerformed

```
uint16 HS_CDSData_t::ResetsPerformed
```

Number of Resets Performed.

Definition at line 68 of file hs_app.h.

Referenced by HS_AppInit(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_SetCDS← Data(), and HS_SetMaxResetsCmd().

11.142.2.4 ResetsPerformedNot

```
uint16 HS_CDSData_t::ResetsPerformedNot
```

Inverted Resets Performed for validation.

Definition at line 69 of file hs_app.h.

Referenced by HS_AppInit(), and HS_SetCDSData().

The documentation for this struct was generated from the following file:

- apps/hs/fsw/src/[hs_app.h](#)

11.143 HS_CustomData_t Struct Reference

HS custom global structure.

```
#include <hs_custom.h>
```

Data Fields

- **int32 UtilMult1**
CPU Utilization Conversion Factor Multiplication 1.
- **int32 UtilDiv**
CPU Utilization Conversion Factor Division.
- **int32 UtilMult2**
CPU Utilization Conversion Factor Multiplication 2.
- **uint32 UtilMask**
Mask for determining Idle Tick length.
- **uint32 UtilArrayIndex**
Index for determining where to write in Util Array.
- **uint32 UtilArrayMask**
Mask for determining where to write in Util Array.
- **uint32 UtilArray [HS_UTIL_TIME_DIAG_ARRAY_LENGTH]**
Array to store time stamps for determining idle tick length.
- **uint32 ThisIdleTaskExec**
Idle Task Exec Counter.
- **uint32 LastIdleTaskExec**
Idle Task Exec Counter at Previous Interval.
- **uint32 LastIdleTaskInterval**
Idle Task Increments during Previous Interval.
- **uint32 UtilCycleCounter**
Counter to determine when to monitor utilization.
- **int32 UtilCallsPerMark**
CPU Utilization Calls per mark.
- **int32 IdleTaskRunStatus**
HS Idle Task Run Status.
- **CFE_ES_TaskId_t IdleTaskID**
HS Idle Task Task ID.

11.143.1 Detailed Description

HS custom global structure.

Definition at line 252 of file hs_custom.h.

11.143.2 Field Documentation

11.143.2.1 IdleTaskID

```
CFE_ES_TaskId_t HS_CustomData_t::IdleTaskID
```

HS Idle Task Task ID.

Definition at line 272 of file hs_custom.h.

Referenced by HS_CustomCleanup(), and HS_CustomInit().

11.143.2.2 IdleTaskRunStatus

```
int32 HS_CustomData_t::IdleTaskRunStatus
```

HS Idle Task Run Status.

Definition at line 271 of file hs_custom.h.

Referenced by HS_CustomCleanup(), HS_CustomInit(), and HS_IdleTask().

11.143.2.3 LastIdleTaskExec

```
uint32 HS_CustomData_t::LastIdleTaskExec
```

Idle Task Exec Counter at Previous Interval.

Definition at line 265 of file hs_custom.h.

Referenced by HS_UtilizationMark().

11.143.2.4 LastIdleTaskInterval

```
uint32 HS_CustomData_t::LastIdleTaskInterval
```

Idle Task Increments during Previous Interval.

Definition at line 266 of file hs_custom.h.

Referenced by HS_CustomGetUtil(), and HS_UtilizationMark().

11.143.2.5 ThisIdleTaskExec

```
uint32 HS_CustomData_t::ThisIdleTaskExec
```

Idle Task Exec Counter.

Definition at line 264 of file hs_custom.h.

Referenced by HS_IdleTask(), HS_UtilizationIncrement(), and HS_UtilizationMark().

11.143.2.6 UtilArray

```
uint32 HS_CustomData_t::UtilArray[HS_UTIL_TIME_DIAG_ARRAY_LENGTH]
```

Array to store time stamps for determining idle tick length.

Definition at line 261 of file hs_custom.h.

Referenced by HS_IdleTask(), and HS_UtilDiagReport().

11.143.2.7 UtilArrayIndex

```
uint32 HS_CustomData_t::UtilArrayIndex
```

Index for determining where to write in Util Array.

Definition at line 259 of file hs_custom.h.

Referenced by HS_IdleTask().

11.143.2.8 UtilArrayMask

```
uint32 HS_CustomData_t::UtilArrayMask
```

Mask for determining where to write in Util Array.

Definition at line 260 of file hs_custom.h.

Referenced by HS_CustomInit(), and HS_IdleTask().

11.143.2.9 UtilCallsPerMark

```
int32 HS_CustomData_t::UtilCallsPerMark
```

CPU Utilization Calls per mark.

Definition at line 269 of file hs_custom.h.

Referenced by HS_CustomInit(), and HS_UtilizationMark().

11.143.2.10 UtilCycleCounter

```
uint32 HS_CustomData_t::UtilCycleCounter
```

Counter to determine when to monitor utilization.

Definition at line 267 of file hs_custom.h.

Referenced by HS_CustomMonitorUtilization().

11.143.2.11 UtilDiv

```
int32 HS_CustomData_t::UtilDiv
```

CPU Utilization Conversion Factor Division.

Definition at line 255 of file hs_custom.h.

Referenced by HS_CustomGetUtil(), HS_CustomInit(), and HS_SetUtilParamsCmd().

11.143.2.12 UtilMask

```
uint32 HS_CustomData_t::UtilMask
```

Mask for determining Idle Tick length.

Definition at line 258 of file hs_custom.h.

Referenced by HS_CustomInit(), HS_IdleTask(), HS_SetUtilDiagCmd(), and HS_UtilDiagReport().

11.143.2.13 UtilMult1

`int32 HS_CustomData_t::UtilMult1`

CPU Utilization Conversion Factor Multiplication 1.

Definition at line 254 of file `hs_custom.h`.

Referenced by `HS_CustomGetUtil()`, `HS_CustomInit()`, and `HS_SetUtilParamsCmd()`.

11.143.2.14 UtilMult2

`int32 HS_CustomData_t::UtilMult2`

CPU Utilization Conversion Factor Multiplication 2.

Definition at line 256 of file `hs_custom.h`.

Referenced by `HS_CustomGetUtil()`, `HS_CustomInit()`, and `HS_SetUtilParamsCmd()`.

The documentation for this struct was generated from the following file:

- `apps/hs/fsw/src/hs_custom.h`

11.144 HS_EMTEEntry_t Struct Reference

Event Monitor Table (EMT) Entry.

```
#include <hs_tbl.h>
```

Data Fields

- `charAppName [OS_MAX_API_NAME]`

Name of application generating event.

- `uint16 NullTerm`

Buffer of nulls to terminate string.

- `uint16 EventID`

Event number of monitored event.

- `uint16 ActionType`

Action to take if event is received.

11.144.1 Detailed Description

Event Monitor Table (EMT) Entry.

Definition at line 77 of file `hs_tbl.h`.

11.144.2 Field Documentation

11.144.2.1 ActionType

```
uint16 HS_EMTEEntry_t::ActionType
```

Action to take if event is received.

Definition at line 82 of file hs_tbl.h.

Referenced by HS_HousekeepingReq(), HS_MonitorEvent(), and HS_ValidateEMTable().

11.144.2.2 AppName

```
char HS_EMTEEntry_t::AppName[OS_MAX_API_NAME]
```

Name of application generating event.

Definition at line 79 of file hs_tbl.h.

Referenced by HS_HousekeepingReq(), and HS_MonitorEvent().

11.144.2.3 EventID

```
uint16 HS_EMTEEntry_t::EventID
```

Event number of monitored event.

Definition at line 81 of file hs_tbl.h.

Referenced by HS_MonitorEvent(), and HS_ValidateEMTable().

11.144.2.4 NullTerm

```
uint16 HS_EMTEEntry_t::NullTerm
```

Buffer of nulls to terminate string.

Definition at line 80 of file hs_tbl.h.

Referenced by HS_ValidateEMTable().

The documentation for this struct was generated from the following file:

- [apps/hs/fsw/src/hs_tbl.h](#)

11.145 HS_HkPacket_t Struct Reference

Housekeeping Packet Structure.

```
#include <hs_msg.h>
```

Data Fields

- **CFE_MSG_TelemetryHeader_t TlmHeader**
Telemetry Header.
- **uint8 CmdCount**
HS Application Command Counter.
- **uint8 CmdErrCount**
HS Application Command Error Counter.
- **uint8 CurrentAppMonState**
Status of HS Application Monitor.
- **uint8 CurrentEventMonState**
Status of HS Event Monitor.
- **uint8 CurrentAlivenessState**
Status of HS Aliveness Indicator.
- **uint8 CurrentCPUHogState**
Status of HS Hogging Indicator.
- **uint8 StatusFlags**
Internal HS Error States.
- **uint8 SpareBytes**
Alignment Spares.
- **uint16 ResetsPerformed**
HS Performed Processor Reset Count.
- **uint16 MaxResets**
HS Maximum Processor Reset Count.
- **uint32 EventsMonitoredCount**
Total count of Event Messages Monitored.
- **uint32 InvalidEventMonCount**
Total count of Invalid Event Monitors.
- **uint32 AppMonEnables [((HS_MAX_MONITORED_APPS - 1)/HS_BITS_PER_APPMON_ENABLE)+1]**
Enable states of App Monitor Entries.
- **uint32 MsgActExec**
Number of Software Bus Message Actions Executed.
- **uint32 UtilCpuAvg**
Current CPU Utilization Average.
- **uint32 UtilCpuPeak**
Current CPU Utilization Peak.

11.145.1 Detailed Description

Housekeeping Packet Structure.

Definition at line 93 of file hs_msg.h.

11.145.2 Field Documentation

11.145.2.1 AppMonEnables

```
uint32 HS_HkPacket_t::AppMonEnables[((HS_MAX_MONITORED_APPS - 1)/HS_BITS_PER_APPMON_ENABLE)+1]
```

Enable states of App Monitor Entries.

Definition at line 110 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.2 CmdCount

```
uint8 HS_HkPacket_t::CmdCount
```

HS Application Command Counter.

Definition at line 97 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.3 CmdErrCount

```
uint8 HS_HkPacket_t::CmdErrCount
```

HS Application Command Error Counter.

Definition at line 98 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.4 CurrentAlivenessState

```
uint8 HS_HkPacket_t::CurrentAlivenessState
```

Status of HS Aliveness Indicator.

Definition at line 101 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.5 CurrentAppMonState

`uint8 HS_HkPacket_t::CurrentAppMonState`

Status of HS Application Monitor.

Definition at line 99 of file `hs_msg.h`.

Referenced by `HS_HousekeepingReq()`.

11.145.2.6 CurrentCPUHogState

`uint8 HS_HkPacket_t::CurrentCPUHogState`

Status of HS Hogging Indicator.

Definition at line 102 of file `hs_msg.h`.

Referenced by `HS_HousekeepingReq()`.

11.145.2.7 CurrentEventMonState

`uint8 HS_HkPacket_t::CurrentEventMonState`

Status of HS Event Monitor.

Definition at line 100 of file `hs_msg.h`.

Referenced by `HS_HousekeepingReq()`.

11.145.2.8 EventsMonitoredCount

`uint32 HS_HkPacket_t::EventsMonitoredCount`

Total count of Event Messages Monitored.

Definition at line 107 of file `hs_msg.h`.

Referenced by `HS_HousekeepingReq()`.

11.145.2.9 InvalidEventMonCount

```
uint32 HS_HkPacket_t::InvalidEventMonCount
```

Total count of Invalid Event Monitors.

Definition at line 108 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.10 MaxResets

```
uint16 HS_HkPacket_t::MaxResets
```

HS Maximum Processor Reset Count.

Definition at line 106 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.11 MsgActExec

```
uint32 HS_HkPacket_t::MsgActExec
```

Number of Software Bus Message Actions Executed.

Definition at line 113 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.12 ResetsPerformed

```
uint16 HS_HkPacket_t::ResetsPerformed
```

HS Performed Processor Reset Count.

Definition at line 105 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.13 SpareBytes

```
uint8 HS_HkPacket_t::SpareBytes
```

Alignment Spares.

Definition at line 104 of file hs_msg.h.

11.145.2.14 StatusFlags

```
uint8 HS_HkPacket_t::StatusFlags
```

Internal HS Error States.

Definition at line 103 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.15 TlmHeader

```
CFE_MSG_TelemetryHeader_t HS_HkPacket_t::TlmHeader
```

Telemetry Header.

Definition at line 95 of file hs_msg.h.

Referenced by HS_HousekeepingReq(), and HS_SblInit().

11.145.2.16 UtilCpuAvg

```
uint32 HS_HkPacket_t::UtilCpuAvg
```

Current CPU Utilization Average.

Definition at line 114 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

11.145.2.17 UtilCpuPeak

```
uint32 HS_HkPacket_t::UtilCpuPeak
```

Current CPU Utilization Peak.

Definition at line 115 of file hs_msg.h.

Referenced by HS_HousekeepingReq().

The documentation for this struct was generated from the following file:

- [apps/hs/fsw/src/hs_msg.h](#)

11.146 HS_MATEEntry_t Struct Reference

Message Actions Table (MAT) Entry.

```
#include <hs_tbl.h>
```

Data Fields

- **uint16 EnableState**
If entry contains message.
- **uint16 Cooldown**
Minimum rate at which message can be sent.
- **HS_MATMsgBuf_t MsgBuf**
Message to be sent.

11.146.1 Detailed Description

Message Actions Table (MAT) Entry.

Definition at line 107 of file hs_tbl.h.

11.146.2 Field Documentation

11.146.2.1 Cooldown

```
uint16 HS_MATEEntry_t::Cooldown
```

Minimum rate at which message can be sent.

Definition at line 110 of file hs_tbl.h.

Referenced by HS_MonitorApplications(), and HS_MonitorEvent().

11.146.2.2 EnableState

`uint16 HS_MATEEntry_t::EnableState`

If entry contains message.

Definition at line 109 of file hs_tbl.h.

Referenced by HS_MonitorApplications(), HS_MonitorEvent(), and HS_ValidateMATable().

11.146.2.3 MsgBuf

`HS_MATMsgBuf_t HS_MATEEntry_t::MsgBuf`

Message to be sent.

Definition at line 111 of file hs_tbl.h.

Referenced by HS_MonitorApplications(), and HS_MonitorEvent().

The documentation for this struct was generated from the following file:

- [apps/hs/fsw/src/hs_tbl.h](#)

11.147 HS_MATMsgBuf_t Union Reference

Message Action Table buffer.

```
#include <hs_tbl.h>
```

Data Fields

- `uint8 Message [HS_MAX_MSG_ACT_SIZE]`
Raw message array for sizing.
- `CFE_SB_Buffer_t Buffer`
Message Buffer for alignment.

11.147.1 Detailed Description

Message Action Table buffer.

Definition at line 98 of file hs_tbl.h.

11.147.2 Field Documentation

11.147.2.1 Buffer

`CFE_SB_Buffer_t HS_MATMsgBuf_t::Buffer`

Message Buffer for alignment.

Definition at line 101 of file hs_tbl.h.

11.147.2.2 Message

`uint8 HS_MATMsgBuf_t::Message[HS_MAX_MSG_ACT_SIZE]`

Raw message array for sizing.

Definition at line 100 of file hs_tbl.h.

The documentation for this union was generated from the following file:

- `apps/hs/fsw/src/hs_tbl.h`

11.148 HS_NoArgsCmd_t Struct Reference

No Arguments Command.

```
#include <hs_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CmdHeader`
Command header.

11.148.1 Detailed Description

No Arguments Command.

For command details see `HS_NOOP_CC`, `HS_RESET_CC`, `HS_ENABLE_APPMON_CC`, `HS_DISABLE_APPMON_CC`, `HS_ENABLE_EVENTMON_CC`, `HS_DISABLE_EVENTMON_CC`, `HS_ENABLE_ALIVENESS_CC`, `HS_DISABLE_ALIVENESS_CC`, `HS_RESET_RESETS_PERFORMED_CC` Also see `HS_SEND_HK_MID`

Definition at line 65 of file hs_msg.h.

11.148.2 Field Documentation

11.148.2.1 CmdHeader

`CFE_MSG_CommandHeader_t HS_NoArgsCmd_t::CmdHeader`

Command header.

Definition at line 67 of file `hs_msg.h`.

The documentation for this struct was generated from the following file:

- `apps/hs/fsw/src/hs_msg.h`

11.149 HS_SetMaxResetsCmd_t Struct Reference

Set Max Resets Command.

```
#include <hs_msg.h>
```

Data Fields

- `CFE_MSG_CommandHeader_t CmdHeader`
Command header.
- `uint16 MaxResets`
Maximum Resets.
- `uint16 Padding`
Structure padding.

11.149.1 Detailed Description

Set Max Resets Command.

For command details see [HS_SET_MAX_RESETS_CC](#)

Definition at line 75 of file `hs_msg.h`.

11.149.2 Field Documentation

11.149.2.1 CmdHeader

`CFE_MSG_CommandHeader_t HS_SetMaxResetsCmd_t::CmdHeader`

Command header.

Definition at line 77 of file hs_msg.h.

11.149.2.2 MaxResets

`uint16 HS_SetMaxResetsCmd_t::MaxResets`

Maximum Resets.

Definition at line 79 of file hs_msg.h.

Referenced by HS_SetMaxResetsCmd().

11.149.2.3 Padding

`uint16 HS_SetMaxResetsCmd_t::Padding`

Structure padding.

Definition at line 80 of file hs_msg.h.

The documentation for this struct was generated from the following file:

- `apps/hs/fsw/src/hs_msg.h`

11.150 HS_SetUtilDiagCmd_t Struct Reference

Set Utilization Diagnostics Command.

`#include <hs_custom.h>`

Data Fields

- `CFE_MSG_CommandHeader_t CmdHeader`
Command header.
- `uint32 Mask`
Utilization Diagnostics Mask.

11.150.1 Detailed Description

Set Utilization Diagnostics Command.

See [HS_SET_UTIL_DIAG_CC](#)

Definition at line 236 of file hs_custom.h.

11.150.2 Field Documentation

11.150.2.1 CmdHeader

`CFE_MSG_CommandHeader_t HS_SetUtilDiagCmd_t::CmdHeader`

Command header.

Definition at line 238 of file hs_custom.h.

11.150.2.2 Mask

`uint32 HS_SetUtilDiagCmd_t::Mask`

Utilization Diagnostics Mask.

Definition at line 240 of file hs_custom.h.

Referenced by `HS_SetUtilDiagCmd()`.

The documentation for this struct was generated from the following file:

- `apps/hs/fsw/src/hs_custom.h`

11.151 HS_SetUtilParamsCmd_t Struct Reference

Set Utililiztion Parameters Command.

```
#include <hs_custom.h>
```

Data Fields

- [CFE_MSG_CommandHeader_t CmdHeader](#)
Command header.
- [int32 Mult1](#)
Multiplier 1 parameter.
- [int32 Div](#)
Divisor parameter.
- [int32 Mult2](#)
Multiplier 2 parameter.

11.151.1 Detailed Description

Set Utililiztion Parameters Command.

See [HS_SET_UTIL_PARAMS_CC](#)

Definition at line 222 of file hs_custom.h.

11.151.2 Field Documentation

11.151.2.1 CmdHeader

[CFE_MSG_CommandHeader_t](#) HS_SetUtilParamsCmd_t::CmdHeader

Command header.

Definition at line 224 of file hs_custom.h.

11.151.2.2 Div

[int32](#) HS_SetUtilParamsCmd_t::Div

Divisor parameter.

Definition at line 227 of file hs_custom.h.

Referenced by [HS_SetUtilParamsCmd\(\)](#).

11.151.2.3 Mult1

```
int32 HS_SetUtilParamsCmd_t::Mult1
```

Multiplier 1 parameter.

Definition at line 226 of file hs_custom.h.

Referenced by HS_SetUtilParamsCmd().

11.151.2.4 Mult2

```
int32 HS_SetUtilParamsCmd_t::Mult2
```

Multiplier 2 parameter.

Definition at line 228 of file hs_custom.h.

Referenced by HS_SetUtilParamsCmd().

The documentation for this struct was generated from the following file:

- apps/hs/fsw/src/[hs_custom.h](#)

11.152 HS_XCTEntry_t Struct Reference

Execution Counters Table (XCT) Entry.

```
#include <hs_tbl.h>
```

Data Fields

- char [ResourceName \[OS_MAX_API_NAME\]](#)
Name of resource being monitored.
- uint32 [NullTerm](#)
Buffer of nulls to terminate string.
- uint32 [ResourceType](#)
Type of execution counter.

11.152.1 Detailed Description

Execution Counters Table (XCT) Entry.

Definition at line 88 of file hs_tbl.h.

11.152.2 Field Documentation

11.152.2.1 NullTerm

```
uint32 HS_XCTEntry_t::NullTerm
```

Buffer of nulls to terminate string.

Definition at line 91 of file hs_tbl.h.

Referenced by HS_ValidateEMTable().

11.152.2.2 ResourceName

```
char HS_XCTEntry_t::ResourceName[OS_MAX_API_NAME]
```

Name of resource being monitored.

Definition at line 90 of file hs_tbl.h.

11.152.2.3 ResourceType

```
uint32 HS_XCTEntry_t::ResourceType
```

Type of execution counter.

Definition at line 92 of file hs_tbl.h.

Referenced by HS_ValidateEMTable().

The documentation for this struct was generated from the following file:

- apps/hs/fsw/src/[hs_tbl.h](#)

11.153 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`

11.153.1 Detailed Description

OSAL binary semaphore properties.

Definition at line 39 of file `osapi-binsem.h`.

11.153.2 Field Documentation

11.153.2.1 `creator`

`osal_id_t OS_bin_sem_prop_t::creator`

Definition at line 42 of file `osapi-binsem.h`.

11.153.2.2 `name`

`char OS_bin_sem_prop_t::name [OS_MAX_API_NAME]`

Definition at line 41 of file `osapi-binsem.h`.

11.153.2.3 `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`

11.154 `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`

11.154.1 Detailed Description

OSAL counting semaphore properties.

Definition at line 32 of file osapi-countsem.h.

11.154.2 Field Documentation**11.154.2.1 creator**

`osal_id_t OS_count_sem_prop_t::creator`

Definition at line 35 of file osapi-countsem.h.

11.154.2.2 name

`char OS_count_sem_prop_t::name [OS_MAX_API_NAME]`

Definition at line 34 of file osapi-countsem.h.

11.154.2.3 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`

11.155 **os_dirent_t Struct Reference**

Directory entry.

```
#include <osapi-dir.h>
```

Data Fields

- char [FileName](#) [[OS_MAX_FILE_NAME](#)]

11.155.1 Detailed Description

Directory entry.

Definition at line 32 of file osapi-dir.h.

11.155.2 Field Documentation

11.155.2.1 FileName

```
char os_dirent_t::FileName[OS\_MAX\_FILE\_NAME]
```

Definition at line 34 of file osapi-dir.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-dir.h](#)

11.156 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]

11.156.1 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 43 of file osapi-select.h.

11.156.2 Field Documentation

11.156.2.1 object_ids

```
uint8 OS_FdSet::object_ids[(OS_MAX_NUM_OPEN_FILES+7)/8]
```

Definition at line 45 of file osapi-select.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-select.h](#)

11.157 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

11.157.1 Detailed Description

OSAL file properties.

Definition at line 49 of file osapi-file.h.

11.157.2 Field Documentation

11.157.2.1 IsValid

```
uint8 OS_file_prop_t::IsValid
```

Definition at line 53 of file osapi-file.h.

11.157.2.2 Path

```
char OS_file_prop_t::Path[OS_MAX_PATH_LEN]
```

Definition at line 51 of file osapi-file.h.

11.157.2.3 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](#)

11.158 os_fsinfo_t Struct Reference

OSAL file system info.

```
#include <osapi-filesystem.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.

11.158.1 Detailed Description

OSAL file system info.

Definition at line 35 of file osapi-filesystem.h.

11.158.2 Field Documentation

11.158.2.1 FreeFds

```
uint32 os_fsinfo_t::FreeFds
```

Total number that are free.

Definition at line 38 of file osapi-filesystems.h.

11.158.2.2 FreeVolumes

```
uint32 os_fsinfo_t::FreeVolumes
```

Total number of volumes free.

Definition at line 40 of file osapi-filesystems.h.

11.158.2.3 MaxFds

```
uint32 os_fsinfo_t::MaxFds
```

Total number of file descriptors.

Definition at line 37 of file osapi-filesystems.h.

11.158.2.4 MaxVolumes

```
uint32 os_fsinfo_t::MaxVolumes
```

Maximum number of volumes.

Definition at line 39 of file osapi-filesystems.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-filesystems.h](#)

11.159 os_fstat_t Struct Reference

File system status.

```
#include <osapi-file.h>
```

Data Fields

- `uint32 FileModeBits`
- `OS_time_t FileTime`
- `size_t FileSize`

11.159.1 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.

11.159.2 Field Documentation

11.159.2.1 FileModeBits

`uint32 os_fstat_t::FileModeBits`

Definition at line 66 of file osapi-file.h.

11.159.2.2 FileSize

`size_t os_fstat_t::FileSize`

Definition at line 68 of file osapi-file.h.

11.159.2.3 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`

11.160 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`

11.160.1 Detailed Description

OSAL heap properties.

See also

[OS_HeapGetInfo\(\)](#)

Definition at line 36 of file osapi-heap.h.

11.160.2 Field Documentation

11.160.2.1 `free_blocks`

`osal_blockcount_t OS_heap_prop_t::free_blocks`

Definition at line 39 of file osapi-heap.h.

11.160.2.2 `free_bytes`

`size_t OS_heap_prop_t::free_bytes`

Definition at line 38 of file osapi-heap.h.

11.160.2.3 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](#)

11.161 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`

11.161.1 Detailed Description

OSAL module address properties.

Definition at line 78 of file osapi-module.h.

11.161.2 Field Documentation

11.161.2.1 bss_address

`cpuaddr OS_module_address_t::bss_address`

Definition at line 86 of file osapi-module.h.

11.161.2.2 bss_size

`cpuaddr OS_module_address_t::bss_size`

Definition at line 87 of file osapi-module.h.

11.161.2.3 code_address

`cpuaddr OS_module_address_t::code_address`

Definition at line 82 of file osapi-module.h.

11.161.2.4 code_size

`cpuaddr OS_module_address_t::code_size`

Definition at line 83 of file osapi-module.h.

11.161.2.5 data_address

`cpuaddr OS_module_address_t::data_address`

Definition at line 84 of file osapi-module.h.

11.161.2.6 data_size

`cpuaddr OS_module_address_t::data_size`

Definition at line 85 of file osapi-module.h.

11.161.2.7 flags

`uint32 OS_module_address_t::flags`

Definition at line 81 of file osapi-module.h.

11.161.2.8 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](#)

11.162 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](#)

11.162.1 Detailed Description

OSAL module properties.

Definition at line 91 of file osapi-module.h.

11.162.2 Field Documentation

11.162.2.1 addr

```
OS_module_address_t OS_module_prop_t::addr
```

Definition at line 97 of file osapi-module.h.

11.162.2.2 entry_point

`cpuaddr OS_module_prop_t::entry_point`

Definition at line 93 of file osapi-module.h.

11.162.2.3 filename

`char OS_module_prop_t::filename[OS_MAX_PATH_LEN]`

Definition at line 95 of file osapi-module.h.

11.162.2.4 host_module_id

`cpuaddr OS_module_prop_t::host_module_id`

Definition at line 94 of file osapi-module.h.

11.162.2.5 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`

11.163 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`

11.163.1 Detailed Description

OSAL mutex properties.

Definition at line 32 of file osapi-mutex.h.

11.163.2 Field Documentation

11.163.2.1 creator

`osal_id_t OS_mut_sem_prop_t::creator`

Definition at line 35 of file osapi-mutex.h.

11.163.2.2 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`

11.164 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`

11.164.1 Detailed Description

OSAL queue properties.

Definition at line 32 of file osapi-queue.h.

11.164.2 Field Documentation

11.164.2.1 creator

```
osal_id_t OS_queue_prop_t::creator
```

Definition at line 35 of file osapi-queue.h.

11.164.2.2 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](#)

11.165 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.

11.165.1 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 109 of file osapi-sockets.h.

11.165.2 Field Documentation

11.165.2.1 ActualLength

`size_t OS_SockAddr_t::ActualLength`

Length of the actual address data.

Definition at line 111 of file osapi-sockets.h.

11.165.2.2 AddrData

`OS_SockAddrData_t OS_SockAddr_t::AddrData`

Abstract Address data.

Definition at line 112 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-sockets.h`

11.166 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.

11.166.1 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 95 of file osapi-sockets.h.

11.166.2 Field Documentation

11.166.2.1 AlignPtr

```
void* OS_SockAddrData_t::AlignPtr
```

Ensures pointer alignment.

Definition at line 99 of file osapi-sockets.h.

11.166.2.2 AlignU32

```
uint32 OS_SockAddrData_t::AlignU32
```

Ensures uint32 alignment.

Definition at line 98 of file osapi-sockets.h.

11.166.2.3 Buffer

```
uint8 OS_SockAddrData_t::Buffer[OS SOCKADDR_MAX_LEN]
```

Ensures length of at least OS SOCKADDR_MAX_LEN.

Definition at line 97 of file osapi-sockets.h.

The documentation for this union was generated from the following file:

- [osal/src/os/inc/osapi-sockets.h](#)

11.167 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.

11.167.1 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 122 of file osapi-sockets.h.

11.167.2 Field Documentation

11.167.2.1 creator

`osal_id_t OS_socket_prop_t::creator`

OSAL TaskID which opened the socket.

Definition at line 125 of file osapi-sockets.h.

11.167.2.2 name

`char OS_socket_prop_t::name[OS_MAX_API_NAME]`

Name of the socket.

Definition at line 124 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-sockets.h`

11.168 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`

11.168.1 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.

11.168.2 Field Documentation

11.168.2.1 Address

```
void(* OS_static_symbol_record_t::Address) (void)
```

Definition at line 116 of file osapi-module.h.

11.168.2.2 Module

```
const char* OS_static_symbol_record_t::Module
```

Definition at line 117 of file osapi-module.h.

11.168.2.3 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](#)

11.169 OS_statvfs_t Struct Reference

```
#include <osapi-filesystem.h>
```

Data Fields

- `size_t block_size`
- `osal_blockcount_t total_blocks`
- `osal_blockcount_t blocks_free`

11.169.1 Detailed Description

Definition at line 49 of file osapi-filesystems.h.

11.169.2 Field Documentation**11.169.2.1 block_size**

`size_t OS_statvfs_t::block_size`

Block size of underlying FS

Definition at line 51 of file osapi-filesystems.h.

11.169.2.2 blocks_free

`osal_blockcount_t OS_statvfs_t::blocks_free`

Available blocks in underlying FS

Definition at line 53 of file osapi-filesystems.h.

11.169.2.3 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`

11.170 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`

11.170.1 Detailed Description

OSAL task properties.

Definition at line 57 of file osapi-task.h.

11.170.2 Field Documentation

11.170.2.1 creator

```
osal_id_t OS_task_prop_t::creator
```

Definition at line 60 of file osapi-task.h.

11.170.2.2 name

```
char OS_task_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 59 of file osapi-task.h.

11.170.2.3 priority

```
osal_priority_t OS_task_prop_t::priority
```

Definition at line 62 of file osapi-task.h.

11.170.2.4 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](#)

11.171 OS_time_t Struct Reference

OSAL time interval structure.

```
#include <osapi-clock.h>
```

Data Fields

- [int64 ticks](#)

11.171.1 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.

11.171.2 Field Documentation

11.171.2.1 ticks

```
int64 OS_time_t::ticks
```

Ticks elapsed since reference point

Definition at line 47 of file osapi-clock.h.

Referenced by HS_IdleTask(), OS_TimeAdd(), OS_TimeAssembleFromMicroseconds(), OS_TimeAssembleFromMilliseconds(), OS_TimeAssembleFromNanoseconds(), OS_TimeAssembleFromSubseconds(), OS_TimeGetFractionalPart(), 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](#)

11.172 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](#)

11.172.1 Detailed Description

Time base properties.

Definition at line 37 of file osapi-timebase.h.

11.172.2 Field Documentation

11.172.2.1 accuracy

```
uint32 OS_timebase_prop_t::accuracy
```

Definition at line 43 of file osapi-timebase.h.

11.172.2.2 creator

```
osal_id_t OS_timebase_prop_t::creator
```

Definition at line 40 of file osapi-timebase.h.

11.172.2.3 freerun_time

```
uint32 OS_timebase_prop_t::freerun_time
```

Definition at line 42 of file osapi-timebase.h.

11.172.2.4 name

```
char OS_timebase_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 39 of file osapi-timebase.h.

11.172.2.5 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](#)

11.173 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](#)

11.173.1 Detailed Description

Timer properties.

Definition at line 37 of file osapi-timer.h.

11.173.2 Field Documentation

11.173.2.1 accuracy

```
uint32 OS_timer_prop_t::accuracy
```

Definition at line 43 of file osapi-timer.h.

11.173.2.2 creator

```
osal_id_t OS_timer_prop_t::creator
```

Definition at line 40 of file osapi-timer.h.

11.173.2.3 interval_time

```
uint32 OS_timer_prop_t::interval_time
```

Definition at line 42 of file osapi-timer.h.

11.173.2.4 name

```
char OS_timer_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 39 of file osapi-timer.h.

11.173.2.5 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/hs/docs/dox_src/cfs_hs.dox File Reference

12.2 apps/hs/fsw/mission_inc/hs_perfids.h File Reference

Macros

- `#define HS_IDLETASK_PERF_ID 37`
Idle task performance ID.
- `#define HS_APPMAIN_PERF_ID 40`
Main task performance ID.

12.2.1 Detailed Description

CFS Health and Safety (HS) Application Performance IDs

12.3 apps/hs/fsw/platform_inc/hs_msgids.h File Reference

Macros

- `#define HS_CMD_MID 0x18AE`
Msg ID for cmds to HS.
- `#define HS_SEND_HK_MID 0x18AF`
Msg ID to request HS housekeeping.
- `#define HS_WAKEUP_MID 0x18B0`
Msg ID to wake up HS.
- `#define HS_HK_TLM_MID 0x08AD`
HS Housekeeping Telemetry.

12.3.1 Detailed Description

CFS Health and Safety (HS) Application Message IDs

12.4 apps/hs/fsw/platform_inc/hs_platform_cfg.h File Reference

Macros

- `#define HS_APP_NAME "HS"`
Application Name.
- `#define HS_IDLE_TASK_NAME "HS_IDLE_TASK"`
Idle Task Configuration Parameters (custom)
- `#define HS_IDLE_TASK_STACK_PTR 0`
- `#define HS_IDLE_TASK_STACK_SIZE 4096`
- `#define HS_IDLE_TASK_FLAGS 0`
- `#define HS_IDLE_TASK_PRIORITY 252`
Idle Task Priority (custom)
- `#define HS_MAX_EXEC_CNT_SLOTS 32`
Maximum reported execution counters.
- `#define HS_MAX_MSG_ACT_TYPES 8`
Maximum message action types.
- `#define HS_MAX_MSG_ACT_SIZE 16`
Maximum message action size (in bytes)
- `#define HS_MAX_MONITORED_APPS 32`
Maximum number of monitored applications.
- `#define HS_MAX_MONITORED_EVENTS 16`
Maximum number of monitored events.
- `#define HS_WATCHDOG_TIMEOUT_VALUE 10000`
Watchdog Timeout Value.
- `#define HS_POST_PROCESSING_DELAY 0`
Time to wait after performing processing (in milliseconds)
- `#define HS_WAKEUP_TIMEOUT 1200`
Wakeup Message Software Bus Timeout.
- `#define HS_CPU_ALIVE_STRING "."`
CPU aliveness output string.
- `#define HS_CPU_ALIVE_PERIOD 5`
CPU aliveness output period.
- `#define HS_MAX_RESTART_ACTIONS 3`
Max Number of Processor Resets that may be performed by HS.
- `#define HS_CMD_PIPE_DEPTH 12`
Software bus command pipe depth.
- `#define HS_EVENT_PIPE_DEPTH 32`
Software bus event pipe depth.
- `#define HS_WAKEUP_PIPE_DEPTH 1`
Software bus wakeup pipe depth.
- `#define HS_RESET_TASK_DELAY 50`
Time to wait before a processor reset (in milliseconds)
- `#define HS_STARTUP_SYNC_TIMEOUT 65000`
Time to wait for all apps to be started (in milliseconds)
- `#define HS_APPMON_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Application Monitor.

- `#define HS_EVENTMON_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Event Monitor.
- `#define HS_ALIVENESS_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the Aliveness Indicator.
- `#define HS_CPUHOG_DEFAULT_STATE HS_STATE_ENABLED`
Default State of the CPU Hogging Indicator.
- `#define HS_AMT_FILENAME "/cf/hs_amt.tbl"`
Application Monitor Table (AMT) filename.
- `#define HS_EMT_FILENAME "/cf/hs_emt.tbl"`
Event Monitor Table (EMT) filename.
- `#define HS_XCT_FILENAME "/cf/hs_xct.tbl"`
Execution Counter Table (XCT) filename.
- `#define HS_MAT_FILENAME "/cf/hs_mat.tbl"`
Message Actions Table (MAT) filename.
- `#define HS_UTIL_CALLS_PER_MARK 1`
CPU Utilization Calls per Mark (custom)
- `#define HS_UTIL_CYCLES_PER_INTERVAL 1`
CPU Utilization Cycles per Interval (custom)
- `#define HS_UTIL_PER_INTERVAL_TOTAL 10000`
CPU Utilization Total Utils Per Interval.
- `#define HS_UTIL_PER_INTERVAL_HOGGING 9900`
CPU Utilization Hogging Utils Per Interval.
- `#define HS_UTIL_CONV_MULT1 1`
CPU Utilization Conversion Factor Multiplication 1 (custom)
- `#define HS_UTIL_CONV_DIV 10000`
CPU Utilization Conversion Factor Division (custom)
- `#define HS_UTIL_CONV_MULT2 1`
CPU Utilization Conversion Factor Multiplication 2 (custom)
- `#define HS_UTIL_HOGGING_TIMEOUT 5`
CPU Utilization Hogging Timeout.
- `#define HS_UTIL_PEAK_NUM_INTERVAL 64`
CPU Peak Utilization Number of Intervals.
- `#define HS_UTIL_AVERAGE_NUM_INTERVAL 4`
CPU Average Utilization Number of Intervals.
- `#define HS_UTIL_DIAG_MASK 0xFFFFFFFF`
CPU Utilization Diagnostics Mask (custom)
- `#define HS_UTIL_TIME_DIAG_ARRAY_POWER 4`
CPU Utilization Diagnostics Array Configuration (custom)
- `#define HS_UTIL_TIME_DIAG_ARRAY_LENGTH (1 << (HS_UTIL_TIME_DIAG_ARRAY_POWER))`
- `#define HS_UTIL_TIME_DIAG_ARRAY_MASK (HS_UTIL_TIME_DIAG_ARRAY_LENGTH - 1)`
- `#define HS_MISSION_REV 0`
Mission specific version number for HS application.

12.4.1 Detailed Description

CFS Health and Safety (HS) Application Platform Configuration Header File

12.5 apps/hs/fsw/src/hs_app.c File Reference

```
#include "hs_app.h"
#include "hs_events.h"
#include "hs_msgids.h"
#include "hs_perfids.h"
#include "hs_monitors.h"
#include "hs_custom.h"
#include "hs_version.h"
#include "hs_cmds.h"
#include "hs_verify.h"
```

Functions

- [void HS_AppMain \(void\)](#)
CFS Health and Safety (HS) application entry point.
- [int32 HS_AppInit \(void\)](#)
Initialize the CFS Health and Safety (HS) application.
- [int32 HS_SbInit \(void\)](#)
Initialize Software Bus.
- [int32 HS_TblInit \(void\)](#)
Initialize cFE Table Services.
- [int32 HS_ProcessMain \(void\)](#)
Perform Normal Periodic Processing.
- [int32 HS_ProcessCommands \(void\)](#)
Process commands received from cFE Software Bus.

Variables

- [HS_AppData_t HS_AppData](#)

12.5.1 Detailed Description

The CFS Health and Safety (HS) provides several utilities including application monitoring, event monitoring, cpu utilization monitoring, aliveness indication, and watchdog servicing.

12.5.2 Function Documentation

12.5.2.1 HS_AppInit()

```
int32 HS_AppInit (
    void )
```

Initialize the CFS Health and Safety (HS) application.

Description

Health and Safety application initialization routine. This function performs all the required startup steps to initialize HS data structures and get the application registered with the cFE services so it can begin to receive command messages.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

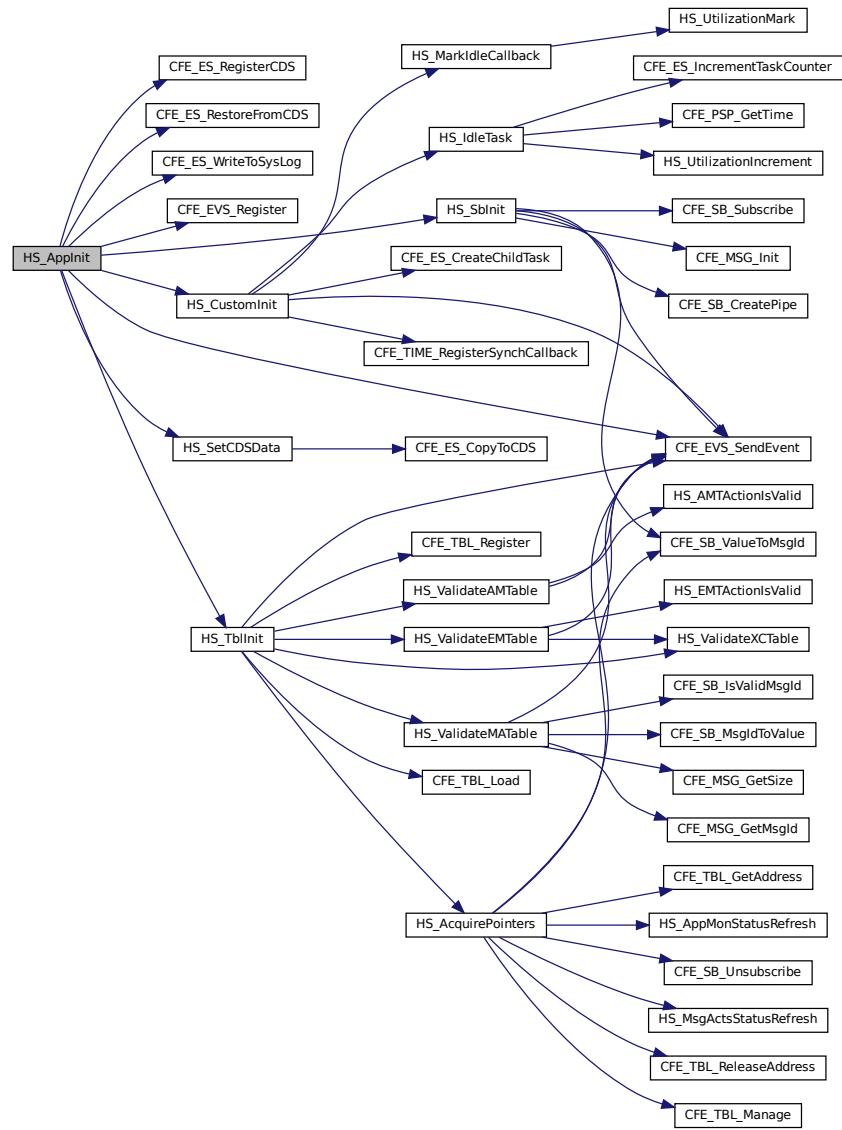
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 203 of file hs_app.c.

References HS_AppData_t::AppMonLoaded, HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_CDS_← ALREADY_EXISTS, CFE_ES_RegisterCDS(), CFE_ES_RestoreFromCDS(), CFE_ES_RunStatus_APP_RUN, CFE_← _ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INF← ORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_SUCCESS, HS_AppData_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentCPUHogState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventMonLoaded, HS_AppData_t::ExeCountState, HS_ALIVENESS_DEFAULT_STATE, HS_APP← MON_DEFAULT_STATE, HS_CDS_CORRUPT_ERR_EID, HS_CDS_RESTORE_ERR_EID, HS_CDSNAME, HS_C← PUHOG_DEFAULT_STATE, HS_CUSTOM_INIT_ERR_EID, HS_CustomInit(), HS_EVENTMON_DEFAULT_STATE, HS_INIT_EID, HS_MAJOR_VERSION, HS_MAX_RESTART_ACTIONS, HS_MINOR_VERSION, HS_MISSION_REV, HS_REVISION, HS_SblInit(), HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_TblInit(), H← S_UTIL_HOGGING_TIMEOUT, HS_AppData_t::MaxCPUHoggingTime, HS_CDSData_t::MaxResets, HS_CDSData← _t::MaxResetsNot, HS_AppData_t::MsgActsState, HS_AppData_t::MyCDSHandle, HS_CDSData_t::ResetsPerformed, HS_CDSData_t::ResetsPerformedNot, HS_AppData_t::RunStatus, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_AppMain().

Here is the call graph for this function:



12.5.2.2 HS_AppMain()

```
void HS_AppMain ( void )
```

CFS Health and Safety (HS) application entry point.

Description

Health and Safety application entry point and main process loop.

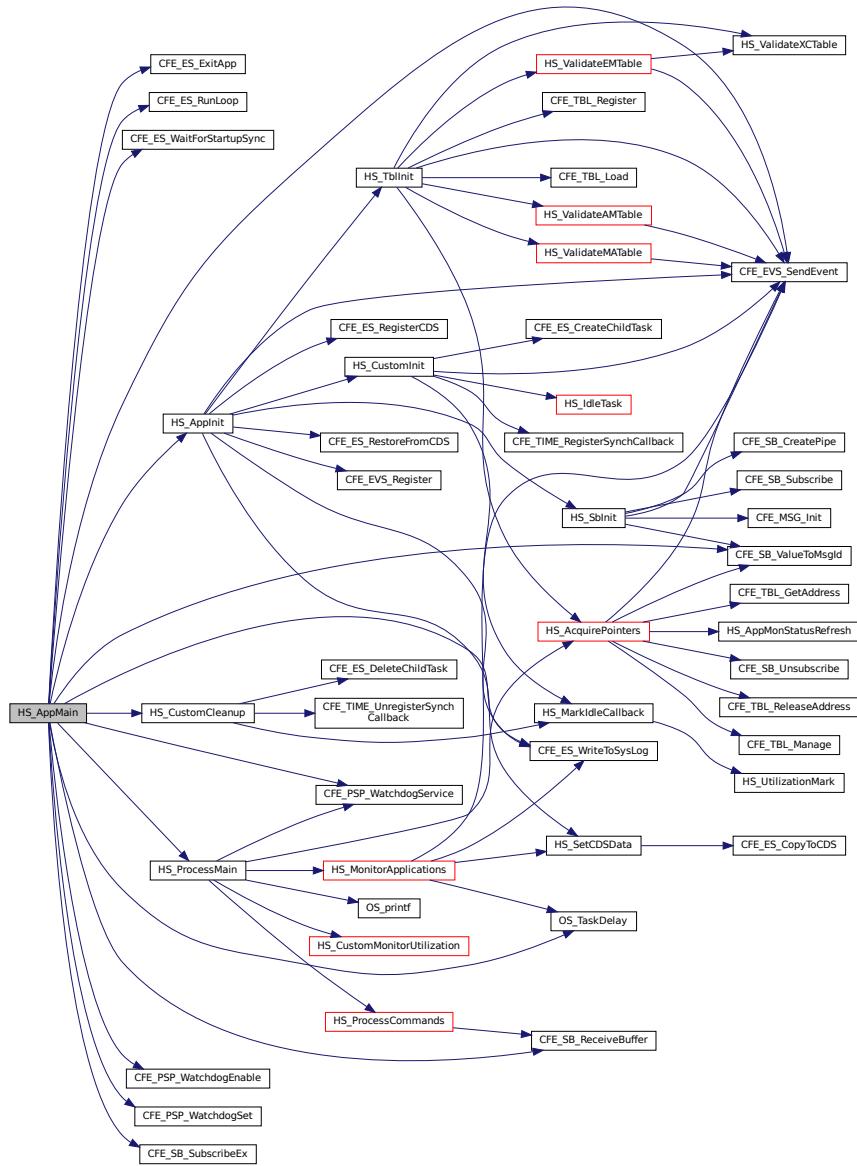
Assumptions, External Events, and Notes:

None

Definition at line 54 of file hs_app.c.

References CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_RUN, CFE_ES_WaitForStartupSync(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_CRITICAL, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_PSP_WatchdogEnable(), CFE_PSP_WatchdogService(), CFE_PSP_WatchdogSet(), CFE_SB_DEFAULT_QOS, CFE_SB_NO_MESSAGE, CFE_SB_ReceiveBuffer(), CFE_SB_SubscribeEx(), CFE_SB_TIME_OUT, CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_APP_EXIT_EID, HS_AppInit(), HS_APPMAIN_PERF_ID, HS_CustomCleanup(), HS_EVENT_PIPE_DEPTH, HS_POST_PROCESSING_DELAY, HS_ProcessMain(), HS_STARTUP_SYNC_TIMEOUT, HS_STATE_ENABLED, HS_SUB_LONG_EVS_ERR_EID, HS_SUB_SHORT_EVS_ERR_EID, HS_WAKEUP_TIMEOUT, HS_WATCHDOG_TIMEOUT_VALUE, OS_TaskDelay(), and HS_AppData_t::WakeupPipe.

Here is the call graph for this function:



12.5.2.3 HS_ProcessCommands()

```
int32 HS_ProcessCommands (
```

Process commands received from cFE Software Bus.

Description

This function pulls messages from command pipe and processes them accordingly.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

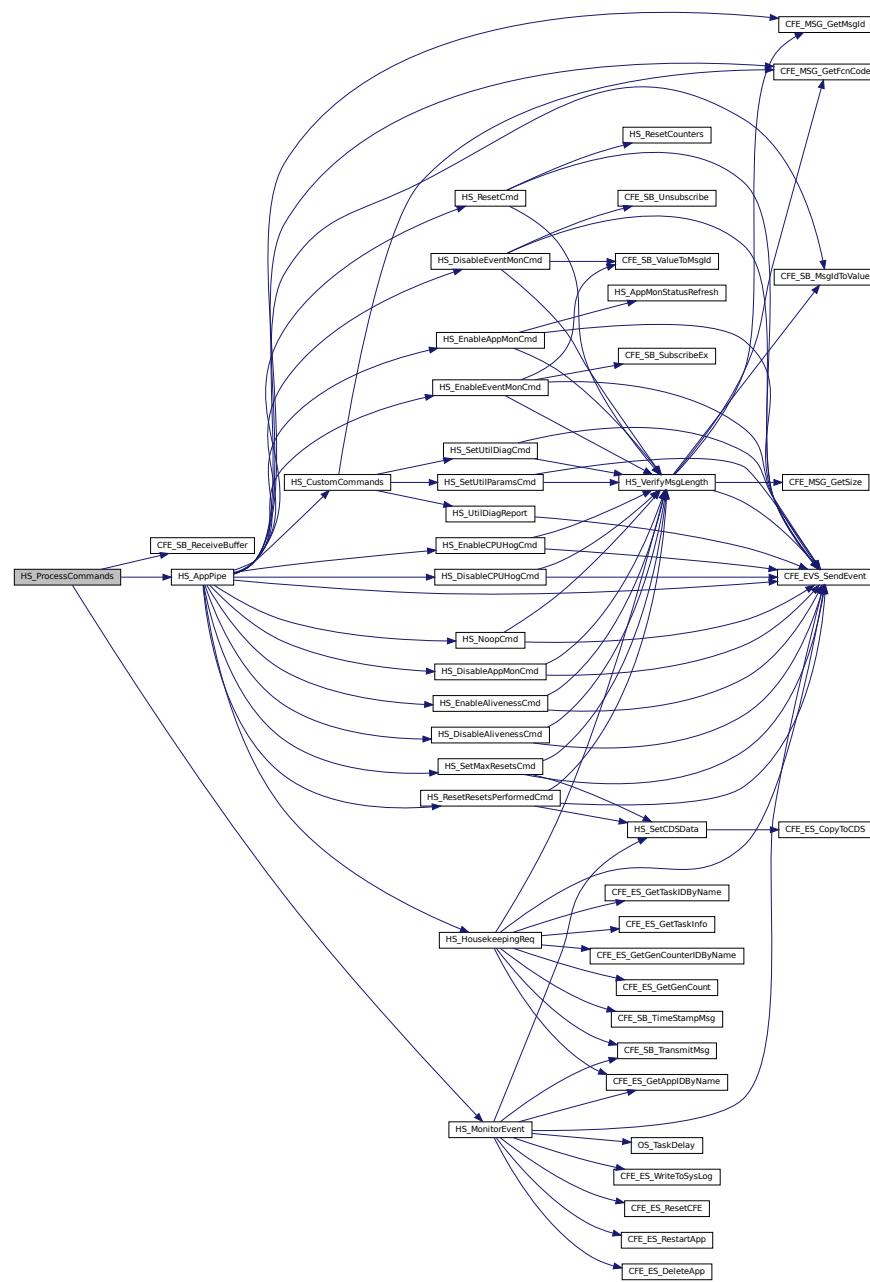
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 613 of file hs_app.c.

References CFE_SB_NO_MESSAGE, CFE_SB_POLL, CFE_SB_ReceiveBuffer(), CFE_SUCCESS, HS_AppData_t::CmdPipe, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData_t::EventsMonitoredCount, HS_AppPipe(), HS_MonitorEvent(), and HS_STATE_ENABLED.

Referenced by HS_ProcessMain().

Here is the call graph for this function:



12.5.2.4 HS_ProcessMain()

```
int32 HS_ProcessMain (
    void )
```

Perform Normal Periodic Processing.

Description

This function performs the normal Health and Safety monitoring functions including application, event and execution counters, as well as servicing the watchdog, outputting the aliveness indicator, and receiving commands or HK requests.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

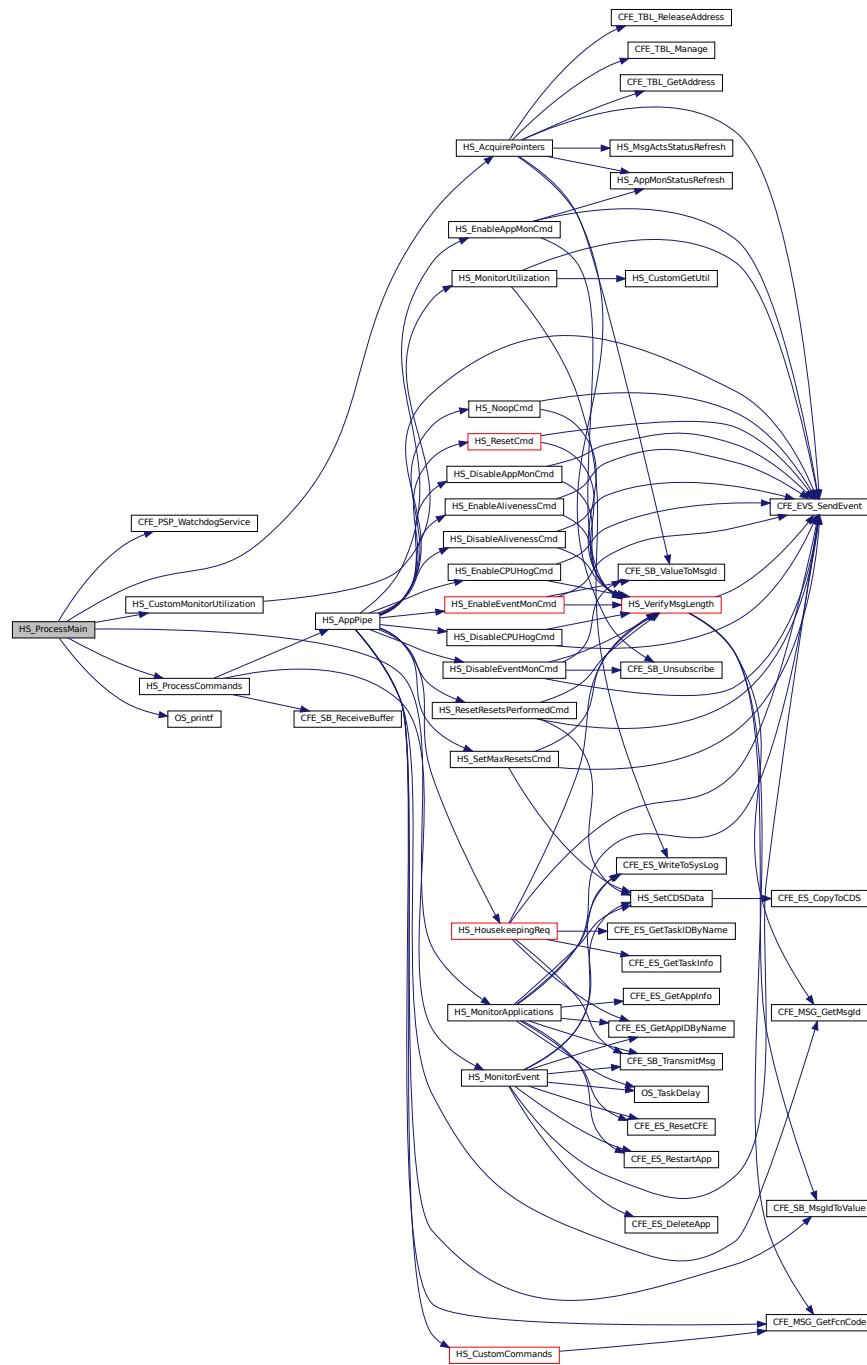
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 542 of file hs_app.c.

References HS_AppData_t::AlivenessCounter, CFE_PSP_WatchdogService(), CFE_SUCCESS, HS_AppData_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_AcquirePointers(), HS_CPU_ALIVE_PERIOD, HS_CPU_ALIVE_STRING, HS_CustomMonitorUtilization(), HS_MAX_MSG_ACT_TYPES, HS_MonitorApplications(), HS_ProcessCommands(), HS_STATE_ENABLED, HS_AppData_t::MsgActCooldown, OS_printf(), and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_AppMain().

Here is the call graph for this function:



12.5.2.5 HS_SbInit()

```
int32 HS_SbInit (
    void )
```

Initialize Software Bus.

Description

This function performs the steps required to setup the cFE software bus for use by the HS application

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

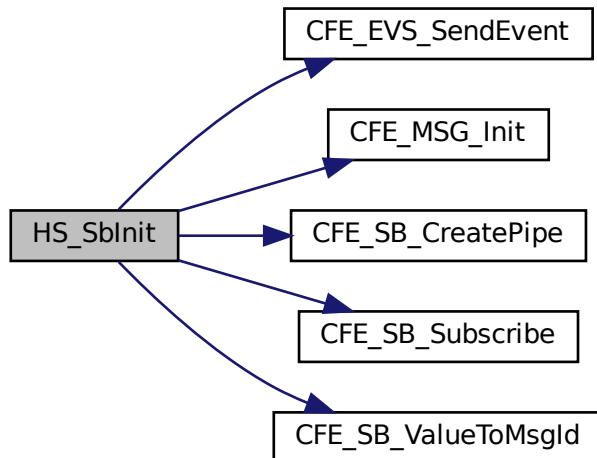
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 351 of file hs_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdPipe, HS_AppData_t::EventPipe, HS_AppData_t::HkPacket, HS_CMD_MID, HS_CMD_PIPE_DEPTH, HS_CMD_PIPE_NAME, HS_CR_CMD_PIPE_ERR_EID, HS_CR_EVENT_PIPE_ERR_EID, HS_CR_WAKEUP_PIPE_ERR_EID, HS_EVENT_PIPE_DEPTH, HS_EVENT_PIPE_NAME, HS_HK_TLM_MID, HS_SEND_HK_MID, HS_SUB_CMD_ERR_EID, HS_SUB_REQ_ERR_EID, HS_SUB_WAKEUP_ERR_EID, HS_WAKEUP_MID, HS_WAKEUP_PIPE_DEPTH, HS_WAKEUP_PIPE_NAME, HS_HkPacket_t::TlmHeader, and HS_AppData_t::WakeUpPipe.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.5.2.6 HS_TblInit()

```
int32 HS_TblInit (
    void )
```

Initialize cFE Table Services.

Description

This function performs those steps required to initialize the relationship between the HS App and the cFE Table Services.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

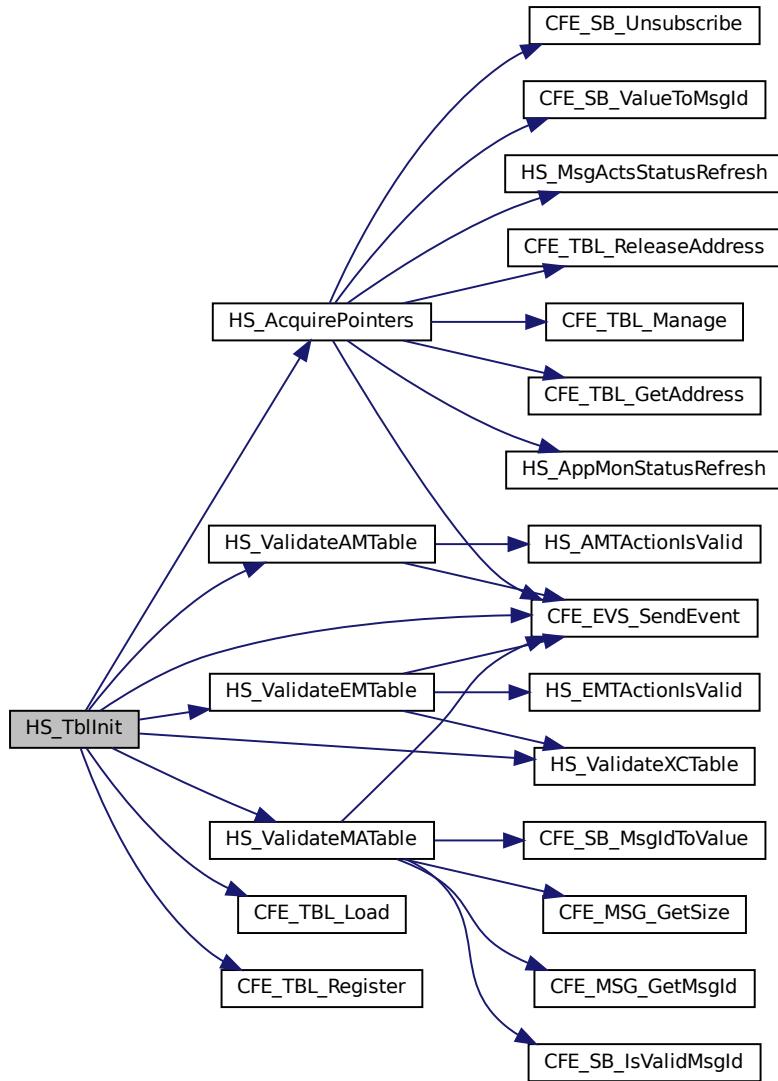
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 425 of file hs_app.c.

References HS_AppData_t::AMTableHandle, HS_AppData_t::AppMonLoaded, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TBL_Load(), CFE_TBL_OPT_DEFAULT, CFE_TBL_Register(), CFE_TBL_SRC_FILE, HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EMTableHandle, HS_AppData_t::EventMonLoaded, HS_AppData_t::ExeCountState, HS_AppData_t::HkPacket, HS_AcquirePointers(), HS_AMT_FILENAME, HS_AMT_LD_ERR_EID, HS_AMT_REG_ERR_EID, HS_AMT_TABLENAME, HS_DISABLE_APPMON_ERR_EID, HS_DISABLE_EVENTMON_ERR_EID, HS_EMT_FILENAME, HS_EMT_LD_ERR_EID, HS_EMT_REG_ERR_EID, HS_EMT_TABLENAME, HS_INVALID_EXECCOUNT, HS_MAT_FILENAME, HS_MAT_LD_ERR_EID, HS_MAT_REG_ERR_EID, HS_MAT_TABLENAME, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_APPS, HS_MAX_MONITORED_EVENTS, HS_MAX_MSG_ACT_TYPES, HS_STATE_DISABLED, HS_ValidateAMTable(), HS_ValidateEMTable(), HS_ValidateMATable(), HS_ValidateXCTable(), HS_XCT_FILENAME, HS_XCT_LD_ERR_EID, HS_XCT_REG_ERR_EID, HS_XCT_TABLENAME, HS_AppData_t::MATableHandle, and HS_AppData_t::MsgActsState.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.5.3 Variable Documentation

12.5.3.1 HS_AppData

`HS_AppData_t HS_AppData`

Definition at line 47 of file `hs_app.c`.

Referenced by HS_AcquirePointers(), HS_AppMonStatusRefresh(), HS_AppPipe(), HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_MonitorUtilization(), HS_MsgActsStatusRefresh(), HS_NoopCmd(), HS_ResetCounters(), HS_ResetResetsPerformedCmd(), HS_SetCDSData(), HS_SetMaxResetsCmd(), HS_SetUtilDiagCmd(), HS_SetUtilParamsCmd(), and HS_VerifyMsgLength().

12.6 apps/hs/fsw/src/hs_app.h File Reference

```
#include "hs_msg.h"
#include "hs_tbl.h"
#include "cfe.h"
#include "cfe_msgids.h"
#include "cfe_platform_cfg.h"
```

Data Structures

- struct [HS_CDSData_t](#)
HS CDS Data Structure.
- struct [HS_AppData_t](#)
HS Global Data Structure.

Macros

- #define [HS_TBL_VAL_ERR](#) (-1)

HS Pipe Parameters

- #define [HS_CMD_PIPE_NAME](#) "HS_CMD_PIPE"
- #define [HS_EVENT_PIPE_NAME](#) "HS_EVENT_PIPE"
- #define [HS_WAKEUP_PIPE_NAME](#) "HS_WAKEUP_PIPE"

HS CDS Buffer String

- #define [HS_CDSNAME](#) "HS_CDS"

Functions

- void [HS_AppMain](#) (void)
CFS Health and Safety (HS) application entry point.
- int32 [HS_ApplInit](#) (void)
Initialize the CFS Health and Safety (HS) application.
- int32 [HS_SblInit](#) (void)
Initialize Software Bus.
- int32 [HS_TblInit](#) (void)
Initialize cFE Table Services.
- int32 [HS_ProcessMain](#) (void)
Perform Normal Periodic Processing.
- int32 [HS_ProcessCommands](#) (void)
Process commands received from cFE Software Bus.

Variables

- [HS_AppData_t HS_AppData](#)

12.6.1 Detailed Description

Unit specification for the Core Flight System (CFS) Health and Safety (HS) Application.

12.6.2 Macro Definition Documentation**12.6.2.1 HS_CDSNAME**

```
#define HS_CDSNAME "HS_CDS"
```

Definition at line 54 of file hs_app.h.

Referenced by HS_AppInit().

12.6.2.2 HS_CMD_PIPE_NAME

```
#define HS_CMD_PIPE_NAME "HS_CMD_PIPE"
```

Definition at line 45 of file hs_app.h.

Referenced by HS_SbInit().

12.6.2.3 HS_EVENT_PIPE_NAME

```
#define HS_EVENT_PIPE_NAME "HS_EVENT_PIPE"
```

Definition at line 46 of file hs_app.h.

Referenced by HS_SbInit().

12.6.2.4 HS_TBL_VAL_ERR

```
#define HS_TBL_VAL_ERR (-1)
```

Definition at line 57 of file hs_app.h.

Referenced by HS_ValidateAMTable(), HS_ValidateEMTable(), and HS_ValidateMATable().

12.6.2.5 HS_WAKEUP_PIPE_NAME

```
#define HS_WAKEUP_PIPE_NAME "HS_WAKEUP_PIPE"
```

Definition at line 47 of file hs_app.h.

Referenced by HS_SbInit().

12.6.3 Function Documentation

12.6.3.1 HS_AppInit()

```
int32 HS_AppInit (
    void )
```

Initialize the CFS Health and Safety (HS) application.

Description

Health and Safety application initialization routine. This function performs all the required startup steps to initialize HS data structures and get the application registered with the cFE services so it can begin to receive command messages.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

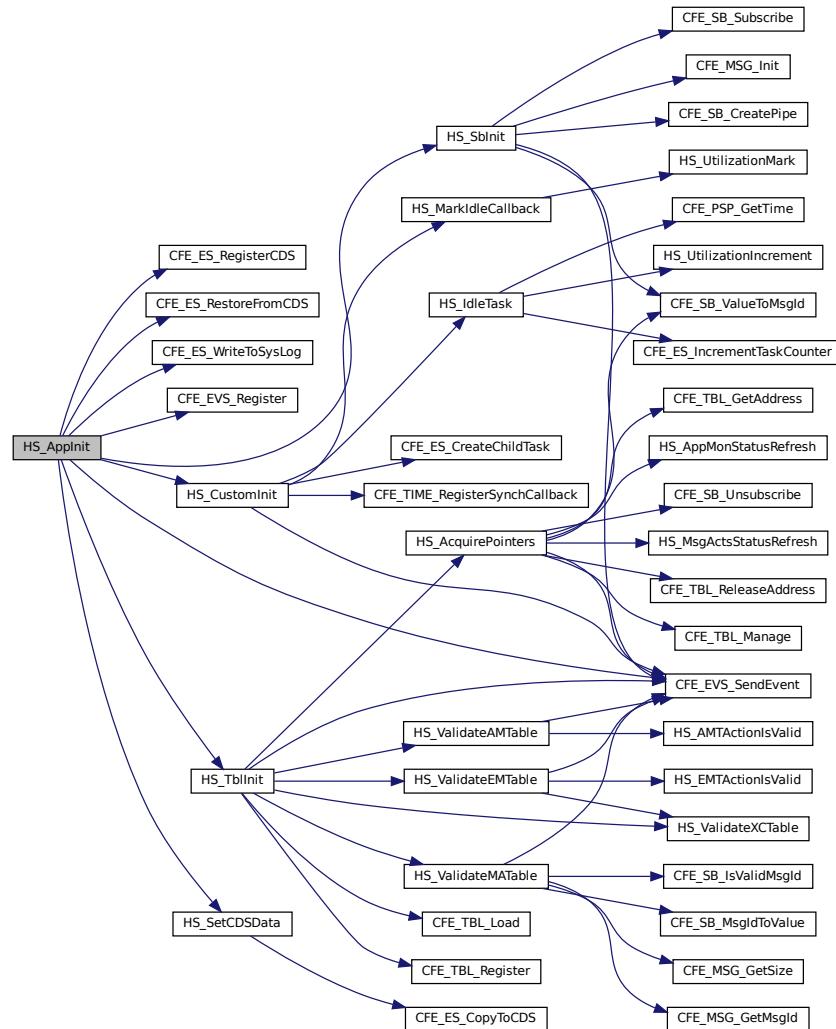
Definition at line 203 of file hs_app.c.

References HS_AppData_t::AppMonLoaded, HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_CDS_← ALREADY_EXISTS, CFE_ES_RegisterCDS(), CFE_ES_RestoreFromCDS(), CFE_ES_RunStatus_APP_RUN, CFE_← _ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INF_← ORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_SUCCESS, HS_AppData_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentCPUHogState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventMonLoaded, HS_AppData_t::ExeCountState, HS_ALIVENESS_DEFAULT_STATE, HS_APP_← MON_DEFAULT_STATE, HS_CDS_CORRUPT_ERR_EID, HS_CDS_RESTORE_ERR_EID, HS_CDSNAME, HS_C_← PUHOG_DEFAULT_STATE, HS_CUSTOM_INIT_ERR_EID, HS_CustomInit(), HS_EVENTMON_DEFAULT_STATE,

HS_INIT_EID, HS_MAJOR_VERSION, HS_MAX_RESTART_ACTIONS, HS_MINOR_VERSION, HS_MISSION_REV, HS_REVISION, HS_SbInit(), HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_TblInit(), HS_UTIL_HOGGING_TIMEOUT, HS_AppData_t::MaxCPUHoggingTime, HS_CDSData_t::MaxResets, HS_CDSData_t::MaxResetsNot, HS_AppData_t::MsgActsState, HS_AppData_t::MyCDSHandle, HS_CDSData_t::ResetsPerformed, HS_CDSData_t::ResetsPerformedNot, HS_AppData_t::RunStatus, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_AppMain().

Here is the call graph for this function:



12.6.3.2 HS_AppMain()

```
void HS_AppMain (
    void )
```

CFS Health and Safety (HS) application entry point.

Description

Health and Safety application entry point and main process loop.

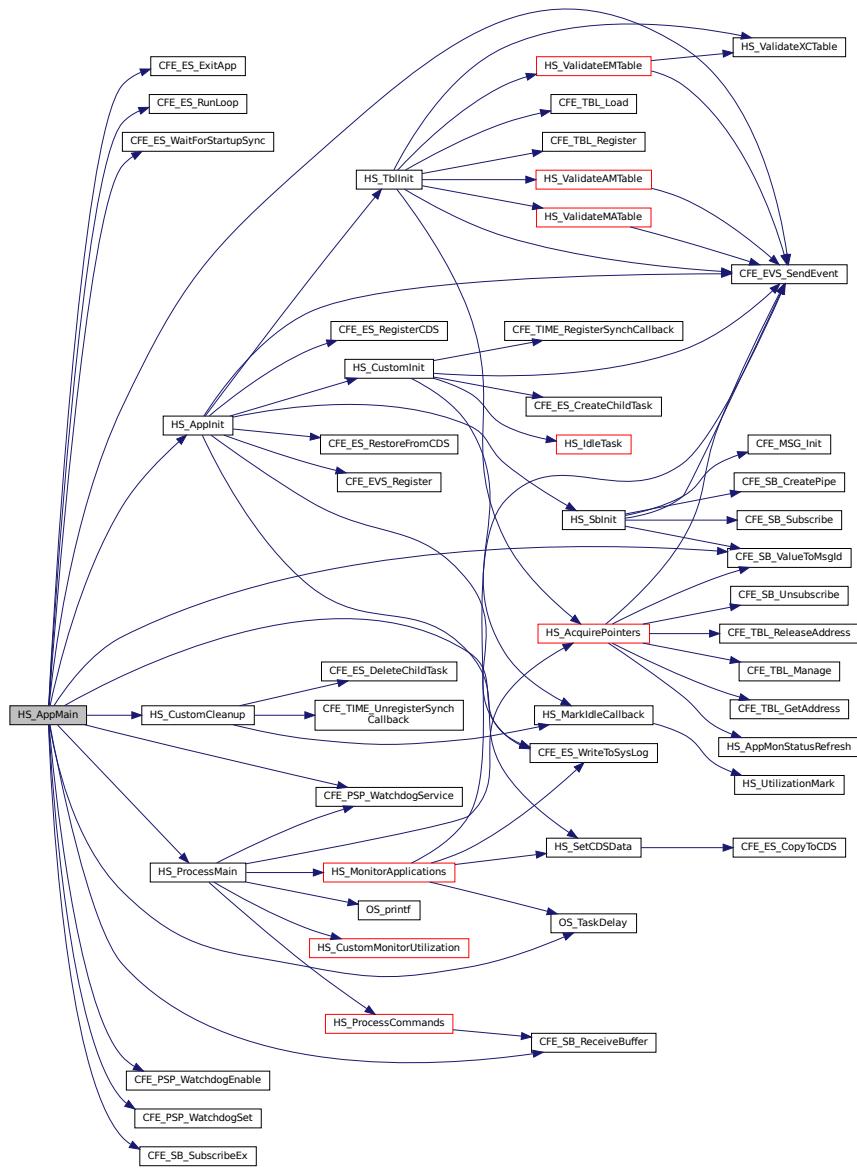
Assumptions, External Events, and Notes:

None

Definition at line 54 of file hs_app.c.

References CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_RUN, CFE_ES_WaitForStartupSync(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_CRITICAL, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_PSP_WatchdogEnable(), CFE_PSP_WatchdogService(), CFE_PSP_WatchdogSet(), CFE_SB_DEFAULT_QOS, CFE_SB_NO_MESSAGE, CFE_SB_ReceiveBuffer(), CFE_SB_SubscribeEx(), CFE_SB_TIME_OUT, CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_APP_EXIT_EID, HS_AppInit(), HS_APPMAIN_PERF_ID, HS_CustomCleanup(), HS_EVENT_PIPE_DEPTH, HS_POST_PROCESSING_DELAY, HS_ProcessMain(), HS_STARTUP_SYNC_TIMEOUT, HS_STATE_ENABLED, HS_SUB_LONG_EVS_ERR_EID, HS_SUB_SHORT_EVS_ERR_EID, HS_WAKEUP_TIMEOUT, HS_WATCHDOG_TIMEOUT_VALUE, OS_TaskDelay(), and HS_AppData_t::WakeupPipe.

Here is the call graph for this function:



12.6.3.3 HS_ProcessCommands()

```
int32 HS_ProcessCommands (
    void )
```

Process commands received from cFE Software Bus.

Description

This function pulls messages from command pipe and processes them accordingly.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

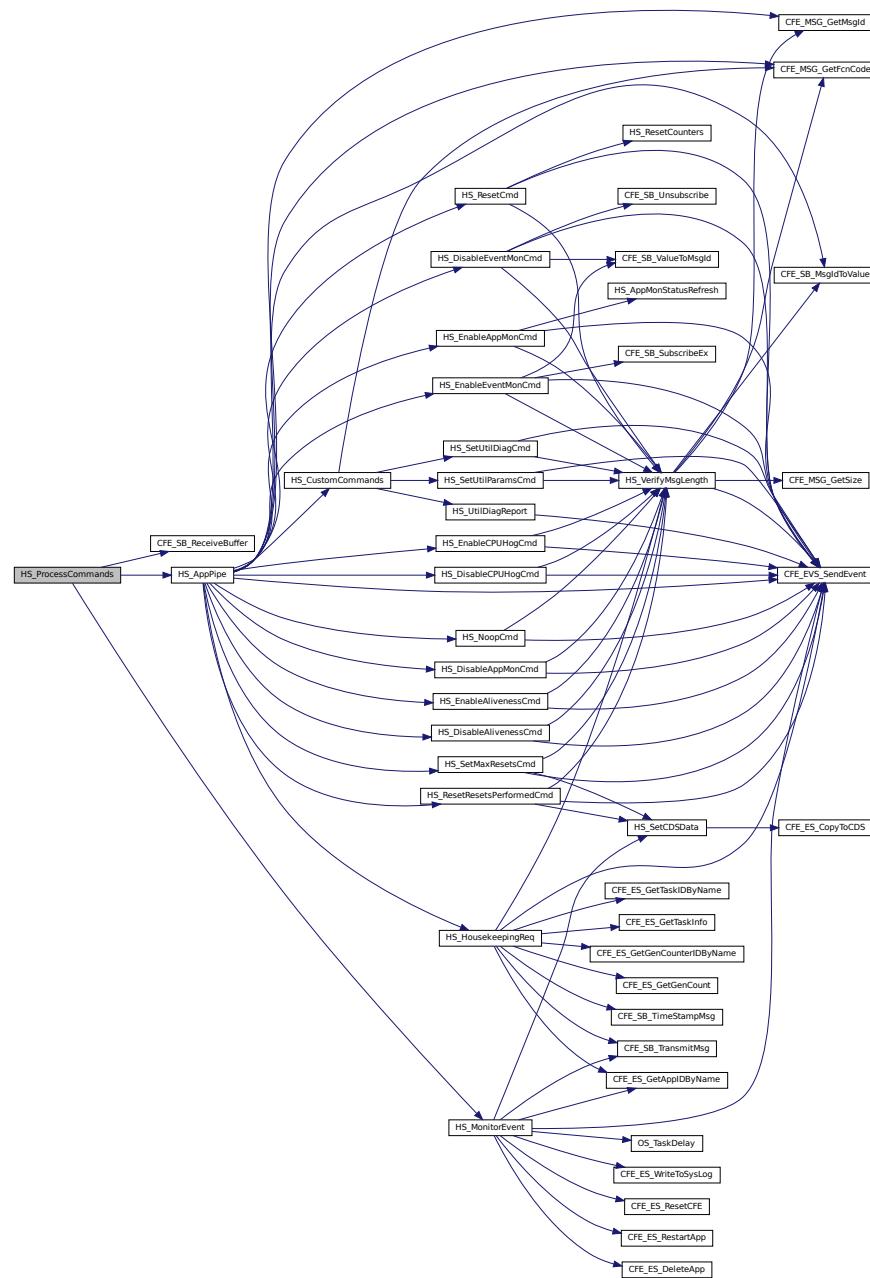
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 613 of file hs_app.c.

References CFE_SB_NO_MESSAGE, CFE_SB_POLL, CFE_SB_ReceiveBuffer(), CFE_SUCCESS, HS_AppData_t::CmdPipe, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData_t::EventsMonitoredCount, HS_AppPipe(), HS_MonitorEvent(), and HS_STATE_ENABLED.

Referenced by HS_ProcessMain().

Here is the call graph for this function:



12.6.3.4 HS_ProcessMain()

```
int32 HS_ProcessMain (
    void )
```

Perform Normal Periodic Processing.

Description

This function performs the normal Health and Safety monitoring functions including application, event and execution counters, as well as servicing the watchdog, outputting the aliveness indicator, and receiving commands or HK requests.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

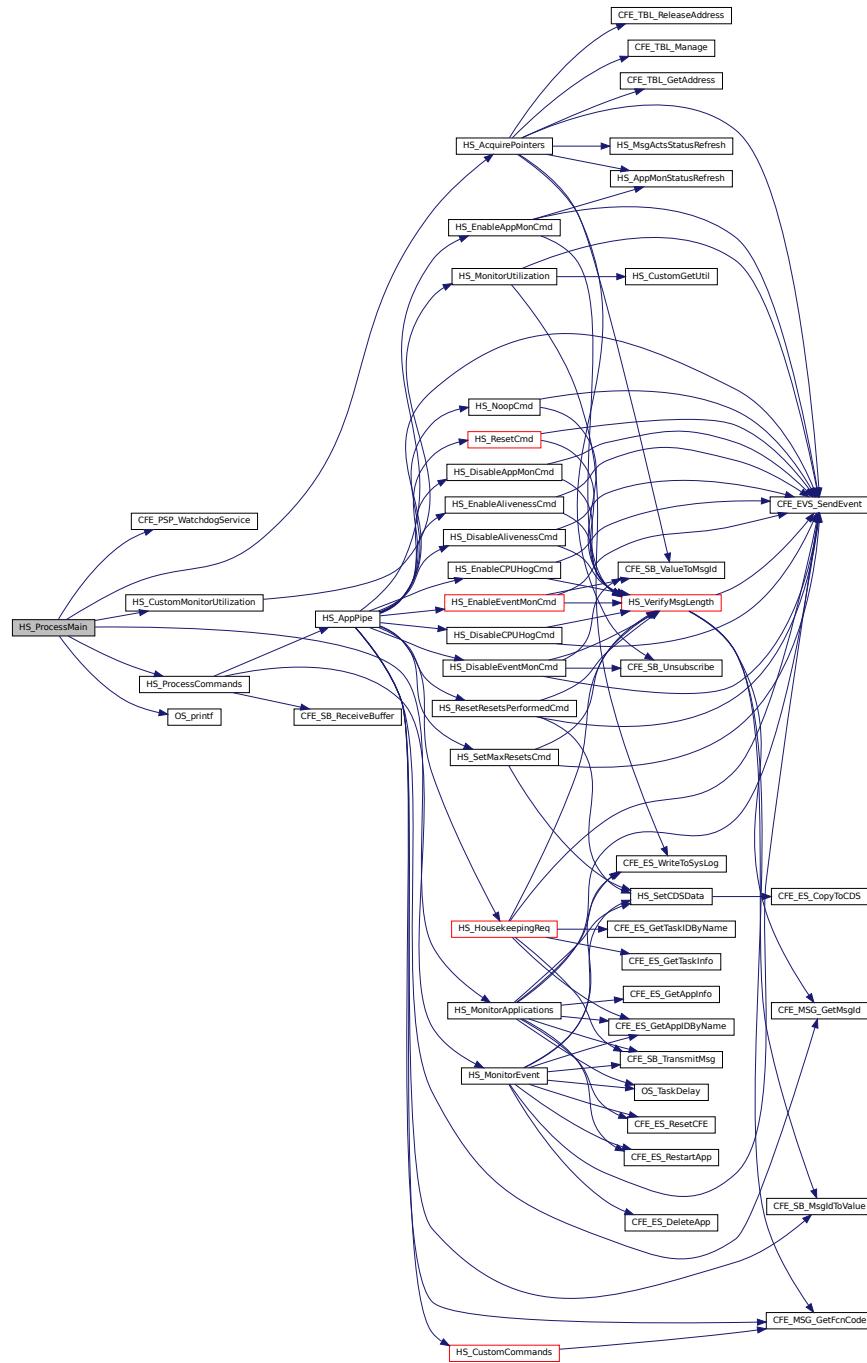
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 542 of file hs_app.c.

References HS_AppData_t::AlivenessCounter, CFE_PSP_WatchdogService(), CFE_SUCCESS, HS_AppData_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_AcquirePointers(), HS_CPU_ALIVE_PERIOD, HS_CPU_ALIVE_STRING, HS_CustomMonitorUtilization(), HS_MAX_MSG_ACT_TYPES, HS_MonitorApplications(), HS_ProcessCommands(), HS_STATE_ENABLED, HS_AppData_t::MsgActCooldown, OS_printf(), and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_AppMain().

Here is the call graph for this function:



12.6.3.5 HS_SblInit()

```
int32 HS_SbInit ( void )
```

Initialize Software Bus.

Description

This function performs the steps required to setup the cFE software bus for use by the HS application

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

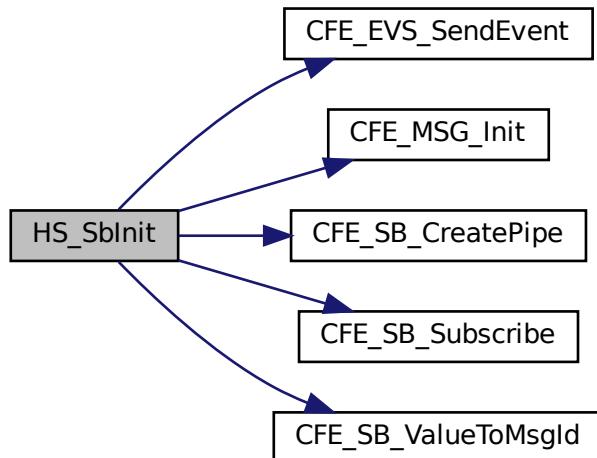
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 351 of file hs_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdPipe, HS_AppData_t::EventPipe, HS_AppData_t::HkPacket, HS_CMD_MID, HS_CMD_PIPE_DEPTH, HS_CMD_PIPE_NAME, HS_CR_CMD_PIPE_ERR_EID, HS_CR_EVENT_PIPE_ERR_EID, HS_CR_WAKEUP_PIPE_ERR_EID, HS_EVENT_PIPE_DEPTH, HS_EVENT_PIPE_NAME, HS_HK_TLM_MID, HS_SEND_HK_MID, HS_SUB_CMD_ERR_EID, HS_SUB_REQ_ERR_EID, HS_SUB_WAKEUP_ERR_EID, HS_WAKEUP_MID, HS_WAKEUP_PIPE_DEPTH, HS_WAKEUP_PIPE_NAME, HS_HkPacket_t::TlmHeader, and HS_AppData_t::WakeUpPipe.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.6.3.6 HS_TblInit()

```
int32 HS_TblInit (
    void )
```

Initialize cFE Table Services.

Description

This function performs those steps required to initialize the relationship between the HS App and the cFE Table Services.

Assumptions, External Events, and Notes:

None

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

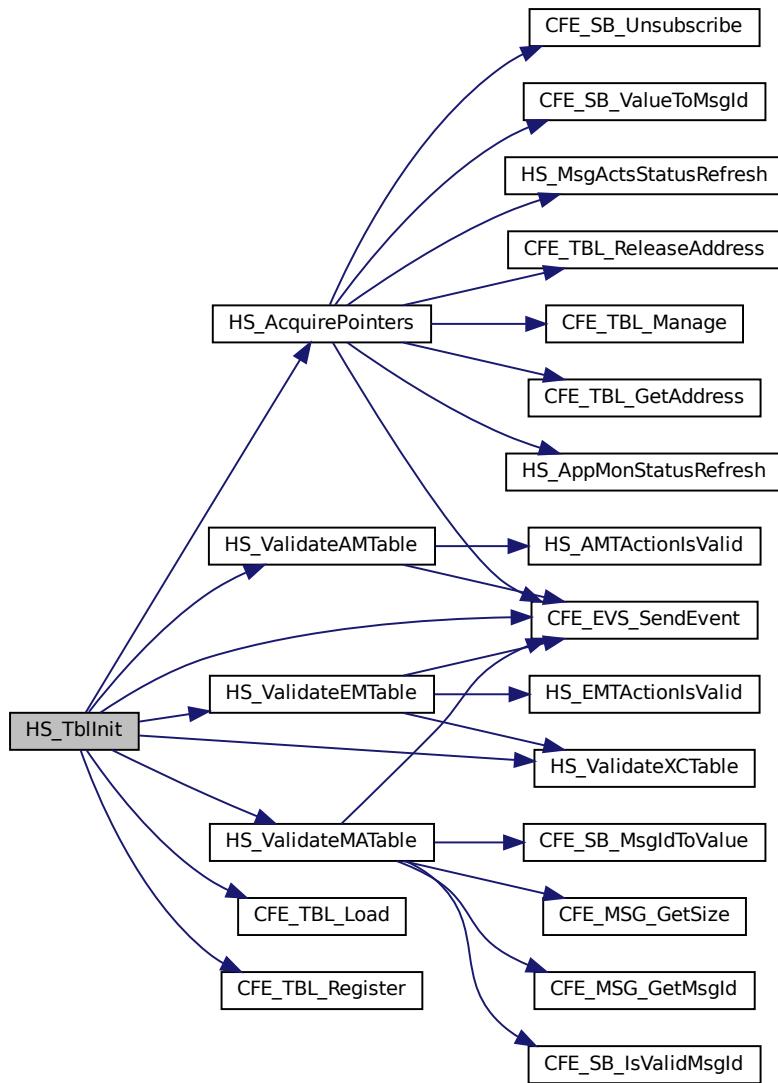
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 425 of file hs_app.c.

References HS_AppData_t::AMTableHandle, HS_AppData_t::AppMonLoaded, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TBL_Load(), CFE_TBL_OPT_DEFAULT, CFE_TBL_Register(), CFE_TBL_SRC_FILE, HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EMTableHandle, HS_AppData_t::EventMonLoaded, HS_AppData_t::ExeCountState, HS_AppData_t::HkPacket, HS_AcquirePointers(), HS_AMT_FILENAME, HS_AMT_LD_ERR_EID, HS_AMT_REG_ERR_EID, HS_AMT_TABLENAME, HS_DISABLE_APPMON_ERR_EID, HS_DISABLE_EVENTMON_ERR_EID, HS_EMT_FILENAME, HS_EMT_LD_ERR_EID, HS_EMT_REG_ERR_EID, HS_EMT_TABLENAME, HS_INVALID_EXECCOUNT, HS_MAT_FILENAME, HS_MAT_LD_ERR_EID, HS_MAT_REG_ERR_EID, HS_MAT_TABLENAME, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_APPS, HS_MAX_MONITORED_EVENTS, HS_MAX_MSG_ACT_TYPES, HS_STATE_DISABLED, HS_ValidateAMTable(), HS_ValidateEMTable(), HS_ValidateMATable(), HS_ValidateXCTable(), HS_XCT_FILENAME, HS_XCT_LD_ERR_EID, HS_XCT_REG_ERR_EID, HS_XCT_TABLENAME, HS_AppData_t::MATableHandle, and HS_AppData_t::MsgActsState.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.6.4 Variable Documentation

12.6.4.1 HS_AppData

`HS_AppData_t HS_AppData`

Definition at line 47 of file `hs_app.c`.

Referenced by HS_AcquirePointers(), HS_AppMonStatusRefresh(), HS_AppPipe(), HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_MonitorUtilization(), HS_MsgActsStatusRefresh(), HS_NoopCmd(), HS_ResetCounters(), HS_ResetResetsPerformedCmd(), HS_SetCDSData(), HS_SetMaxResetsCmd(), HS_SetUtilDiagCmd(), HS_SetUtilParamsCmd(), and HS_VerifyMsgLength().

12.7 apps/hs/fsw/src/hs_cmds.c File Reference

```
#include "hs_app.h"
#include "hs_cmds.h"
#include "hs_custom.h"
#include "hs_monitors.h"
#include "hs_msgids.h"
#include "hs_events.h"
#include "hs_utils.h"
#include "hs_version.h"
```

Functions

- void **HS_AppPipe** (const **CFE_SB_Buffer_t** *BufPtr)
Process a command pipe message.
- void **HS_HousekeepingReq** (const **CFE_SB_Buffer_t** *BufPtr)
Housekeeping request.
- void **HS_NoopCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Noop command.
- void **HS_ResetCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Reset counters command.
- void **HS_ResetCounters** (void)
Reset counters.
- void **HS_EnableAppMonCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process an enable critical applications monitor command.
- void **HS_DisableAppMonCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process a disable critical applications monitor command.
- void **HS_EnableEventMonCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process an enable critical events monitor command.
- void **HS_DisableEventMonCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process a disable critical events monitor command.
- void **HS_EnableAlivenessCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process an enable aliveness indicator command.
- void **HS_DisableAlivenessCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process a disable aliveness indicator command.
- void **HS_EnableCPUHogCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process an enable CPU Hogging indicator command.
- void **HS_DisableCPUHogCmd** (const **CFE_SB_Buffer_t** *BufPtr)
Process a disable CPU Hogging indicator command.

- void [HS_ResetResetsPerformedCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a reset resets performed command.
- void [HS_SetMaxResetsCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a set max resets command.
- void [HS_AcquirePointers](#) (void)
Manages HS tables.
- void [HS_AppMonStatusRefresh](#) (void)
Refresh Critical Applications Monitor Status.
- void [HS_MsgActsStatusRefresh](#) (void)
Refresh Message Actions Status.

12.7.1 Detailed Description

CFS Health and Safety (HS) command handling routines

12.7.2 Function Documentation

12.7.2.1 [HS_AcquirePointers\(\)](#)

```
void HS_AcquirePointers (
    void )
```

Manages HS tables.

Description

Manages load requests for the AppMon, EventMon, ExeCount and MsgActs tables and update notification for the AppMon and MsgActs tables. Also releases and acquires table addresses. Gets called at the start of each processing cycle and on initialization.

Assumptions, External Events, and Notes:

None

See also

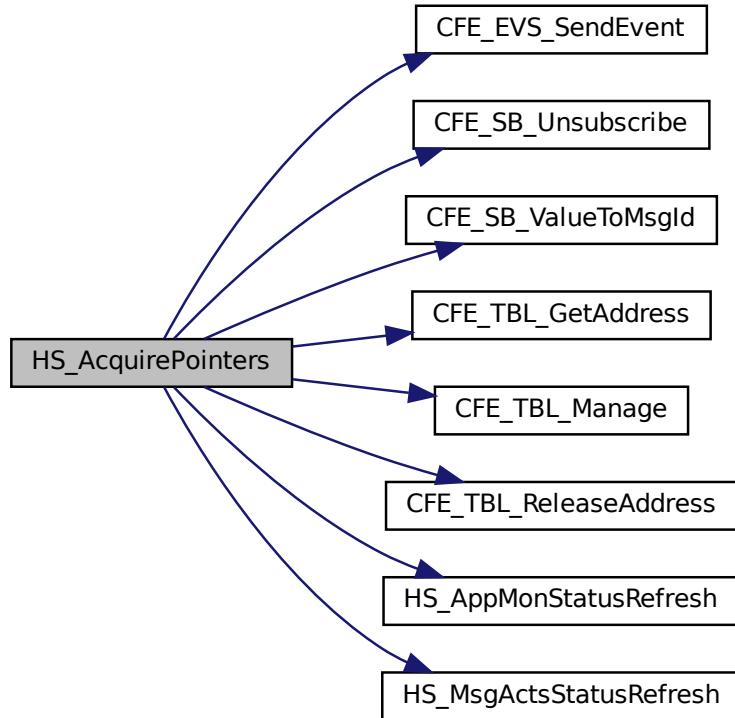
[CFE_TBL_Manage](#)

Definition at line 684 of file hs_cmds.c.

References HS_AppData_t::AMTableHandle, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonLoaded, CFE_EVS_SEND_EVENT_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_Unsubscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CFE_TBL_GetAddress(), CFE_TBL_INFO_UPDATED, CFE_TBL_Manage(), CFE_TBL_ReleaseAddress(), HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EMTableHandle, HS_AppData_t::EMTablePtr, HS_AppData_t::EventMonLoaded, HS_AppData_t::EventPipe, HS_AppData_t::ExeCountState, HS_AppData, HS_APPMON_GETHADDR_ERR_EID, HS_AppMonStatusRefresh(), HS_BADEMT_LONG_UNSUB_EID, HS_BADEMT_SHORT_UNSUB_EID, HS_EVENTMON_GETADDR_ERR_EID, HS_EXECOUNT_GETADDR_ERR_EID, HS_MSGACTS_GETADDR_ERR_EID, HS_MsgActsStatusRefresh(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATableHandle, HS_AppData_t::MATablePtr, and HS_AppData_t::MsgActsState.

Referenced by HS_ProcessMain(), and HS_TblInit().

Here is the call graph for this function:



12.7.2.2 HS_AppMonStatusRefresh()

```
void HS_AppMonStatusRefresh (
    void )
```

Refresh Critical Applications Monitor Status.

Description

This function gets called when HS detects that a new critical applications monitor table has been loaded or when a command to enable the critical applications monitor is received: it then refreshes the timeouts for application being monitored

Assumptions, External Events, and Notes:

None

Definition at line 892 of file hs_cmds.c.

References HS_AMTEntry_t::ActionType, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonCheckInCountdown, HS_AppData_t::AppMonEnables, HS_AppData_t::AppMonLastExeCount, CFE_SET, HS_AMTEntry_t::CycleCount, HS_AMT_ACT_NOACT, HS_AppData, HS_BITS_PER_APPMON_ENABLE, and HS_MAX_MONITORED_APPS.

Referenced by HS_AcquirePointers(), and HS_EnableAppMonCmd().

12.7.2.3 HS_AppPipe()

```
void HS_AppPipe (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a command pipe message.

Description

Processes a single software bus command pipe message. Checks the message and command IDs and calls the appropriate routine to handle the message.

Assumptions, External Events, and Notes:

None

Parameters

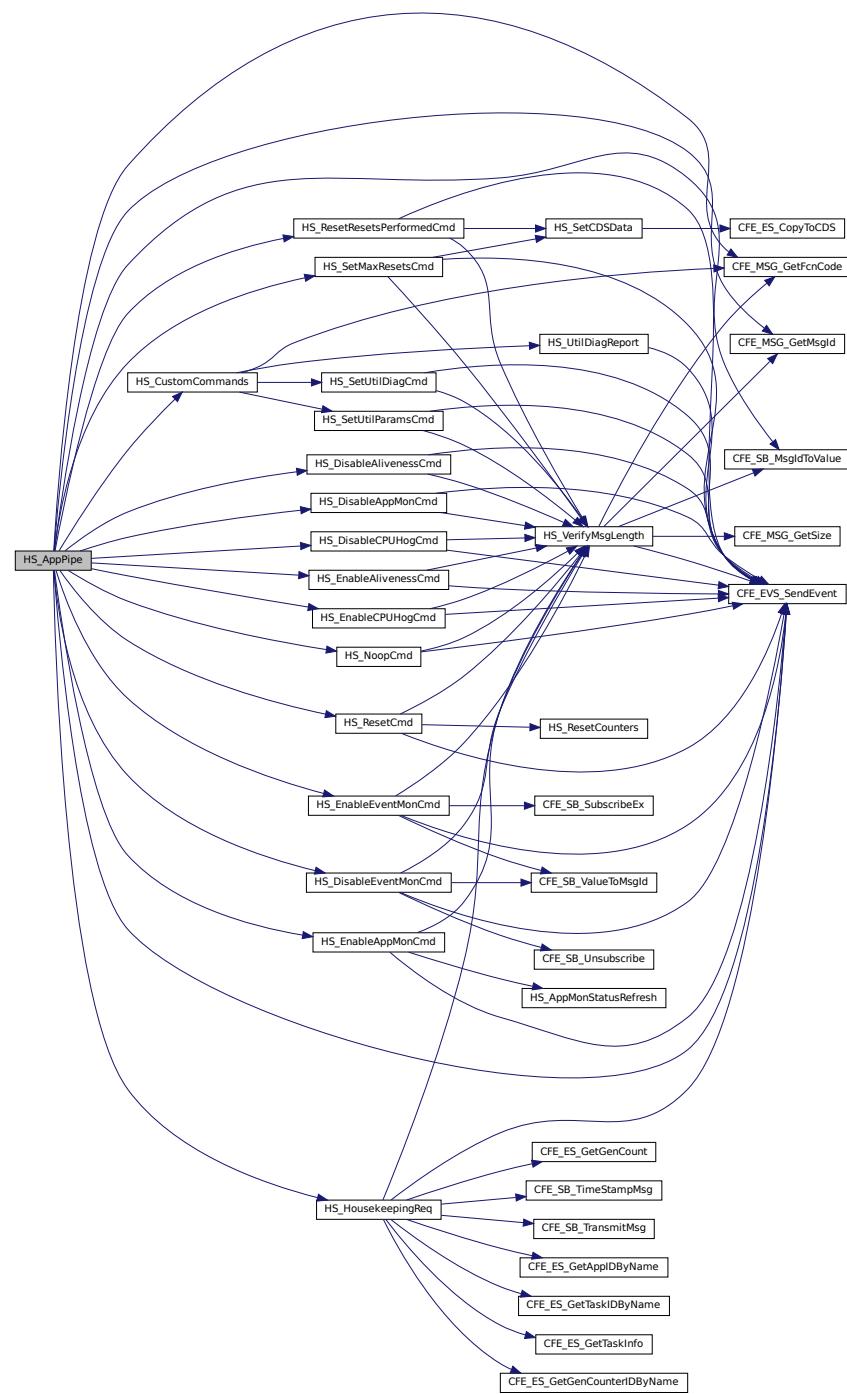
in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

Definition at line 42 of file hs_cmds.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG_GetMsgId(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), CFE_SUCCESS, HS_AppData_t::CmdErrCount, HS_AppData, HS_CC_ERR_EID, HS_CMD_MID, HS_CustomCommands(), HS_DISABLE_ALIVENESS_CC, HS_DisableAppMonCC, HS_DisableCPUHogCC, HS_DisableEventMonCC, HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_ENABLE_ALIVENESS_CC, HS_ENABLE_APPMON_CC, HS_ENABLE_CPUHOG_CC, HS_ENABLE_EVENTMON_CC, HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_MID_ERR_EID, HS_NOOP_CC, HS_NoopCmd(), HS_RESET_CC, HS_RESET_RESETS_PERFORMED_CC, HS_ResetCmd(), HS_ResetResetsPerformedCmd(), HS_SEND_HK_MID, HS_SET_MAX_RESETS_CC, HS_SetMaxResetsCmd(), and CFE_SB_Msg::Msg.

Referenced by HS_ProcessCommands().

Here is the call graph for this function:



12.7.2.4 HS_DisableAlivenessCmd()

```
void HS_DisableAlivenessCmd (
```

```
const CFE_SB_Buffer_t * BufPtr )
```

Process a disable aliveness indicator command.

Description

Stops the aliveness indicator from being output on the UART.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

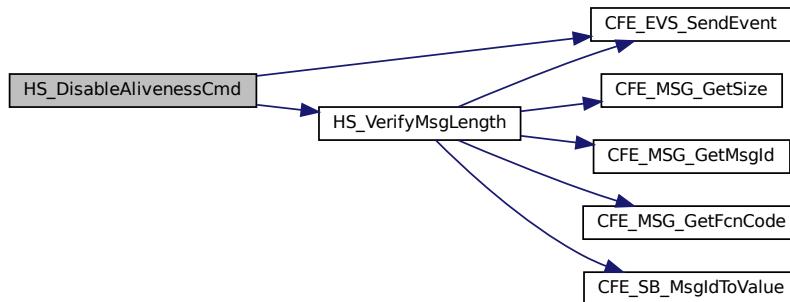
[HS_DISABLE_ALIVENESS_CC](#)

Definition at line 563 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAlivenessState, HS_AppData, HS_DISABLE_ALIVENESS_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.5 HS_DisableAppMonCmd()

```
void HS_DisableAppMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable critical applications monitor command.

Description

Stops the critical applications from be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

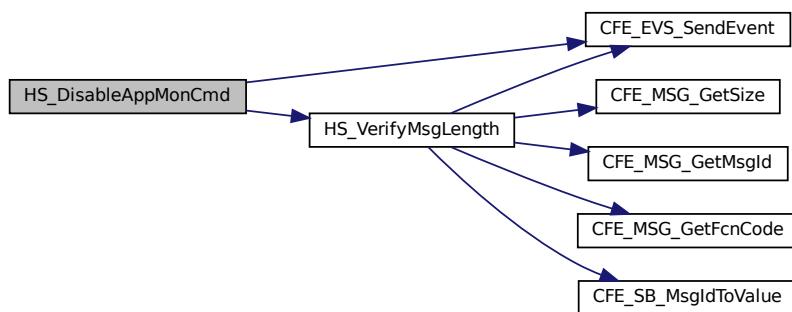
[HS_DISABLE_APPMON_CC](#)

Definition at line 396 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAppMonState, HS_AppData, HS_DISABLE_APPMON_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.6 HS_DisableCPUHogCmd()

```
void HS_DisableCPUHogCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable CPU Hogging indicator command.

Description

Stops the CPU Hogging indicator from being output as an event.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

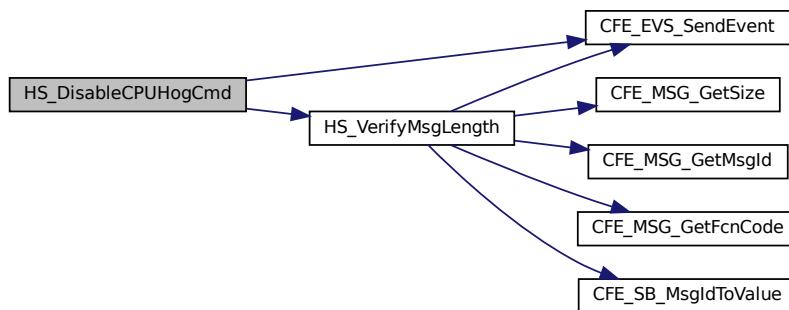
[HS_DISABLE_CPUHOG_CC](#)

Definition at line 609 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentCPUHogState, HS_AppData, HS_DISABLE_CPUHOG_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.7 HS_DisableEventMonCmd()

```
void HS_DisableEventMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable critical events monitor command.

Description

Stops the critical events from be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

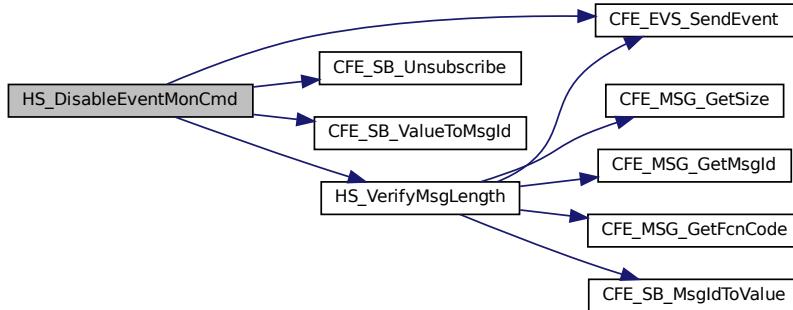
[HS_DISABLE_EVENTMON_CC](#)

Definition at line 480 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_Unsubscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData, HS_DISABLE_EVENTMON_DBG_EID, HS_EVENTMON_LONG_UNSUB_EID, HS_EVENTMON_SHORT_UNSUB_EID, HS_STATE_DISABLED, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.8 HS_EnableAlivenessCmd()

```
void HS_EnableAlivenessCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable aliveness indicator command.

Description

Allows the aliveness indicator to be output to the UART.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

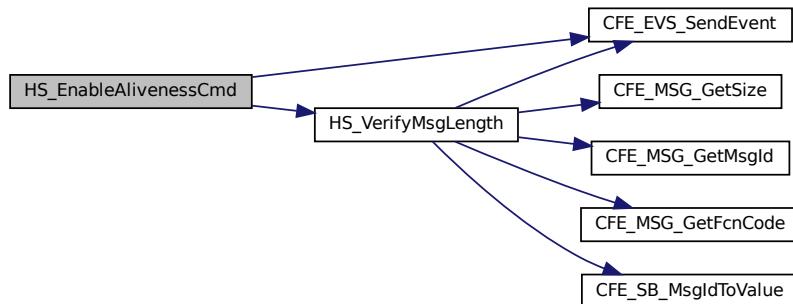
[HS_ENABLE_ALIVENESS_CC](#)

Definition at line 540 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAlivenessState, HS_AppData, HS_ENABLE_ALIVENESS_DBG_EID, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.9 HS_EnableAppMonCmd()

```
void HS_EnableAppMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable critical applications monitor command.

Description

Allows the critical applications to be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

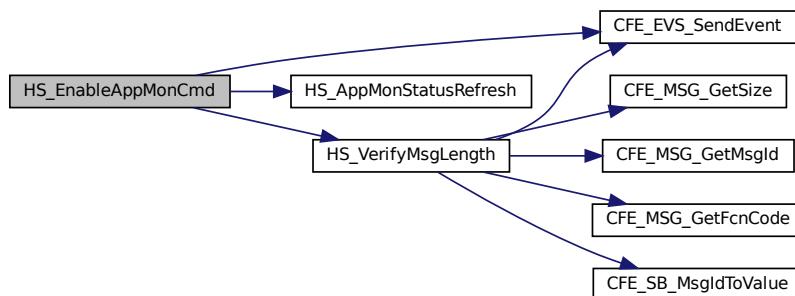
[HS_ENABLE_APPMON_CC](#)

Definition at line 372 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAppMonState, HS_AppData, HS_AppMonStatusRefresh(), HS_ENABLE_APPMON_DBG_EID, HS_STATEENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.10 HS_EnableCPUHogCmd()

```
void HS_EnableCPUHogCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable CPU Hogging indicator command.

Description

Allows the CPU Hogging indicator to be output as an event.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

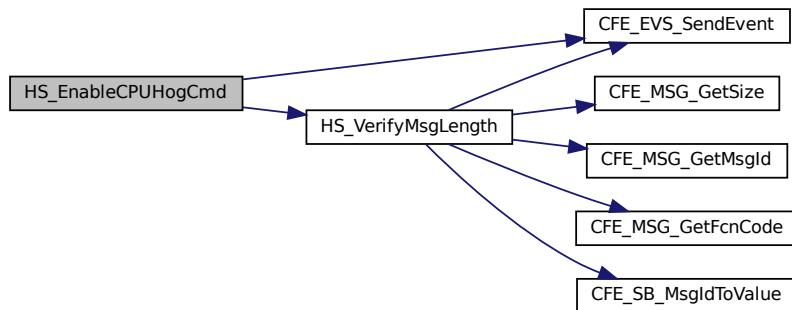
[HS_ENABLE_CPUHOG_CC](#)

Definition at line 586 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentCPUHogState, HS_AppData, HS_ENABLE_CPUHOG_DBG_EID, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.11 HS_EnableEventMonCmd()

```
void HS_EnableEventMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable critical events monitor command.

Description

Allows the critical events to be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

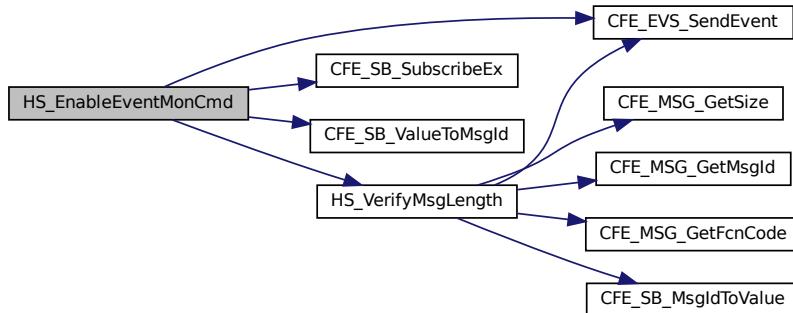
[HS_ENABLE_EVENTMON_CC](#)

Definition at line 419 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_DEFAULT_QOS, CFE_SB_SubscribeEx(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData, HS_ENABLE_EVENTMON_DBG_EID, HS_EVENT_PIPE_DEPTH, HS_EVENTMON_LONG_SUB_EID, HS_EVENTMON_SHORT_SUB_EID, HS_STATE_DISABLED, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.12 HS_HousekeepingReq()

```
void HS_HousekeepingReq (
    const CFE_SB_Buffer_t * BufPtr )
```

Housekeeping request.

Description

Processes an on-board housekeeping request message.

Assumptions, External Events, and Notes:

This message does not affect the command execution counter

Parameters

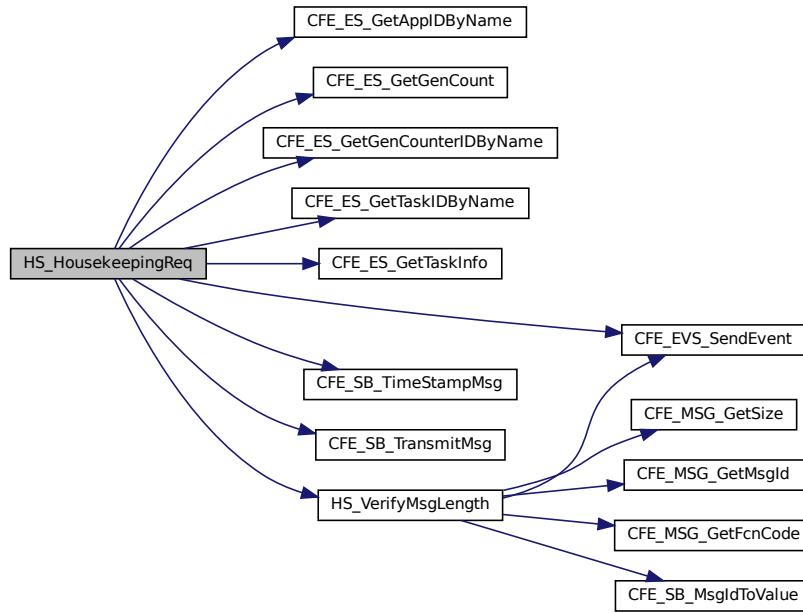
in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

Definition at line 150 of file hs_cmds.c.

References HS_EMTEEntry_t::ActionType, HS_AppData_t::AppMonEnables, HS_HkPacket_t::AppMonEnables, HS_AppData_t::AppMonLoaded, HS_EMTEEntry_t::AppName, HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_APPID_UNDEFINED, CFE_ES_COUNTERID_UNDEFINED, CFE_ES_GetAppIDByName(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CFE_ES_GetTaskIDByName(), CFE_ES_GetTaskInfo(), CFE_ES_TASKID_UNDEFINED, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_HkPacket_t::CmdCount, HS_AppData_t::CmdCount, HS_HkPacket_t::CmdErrCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentAlivenessState, HS_HkPacket_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_HkPacket_t::CurrentAppMonState, HS_AppData_t::CurrentCPUHogState, HS_HkPacket_t::CurrentCPUHogState, HS_AppData_t::CurrentEventMonState, HS_HkPacket_t::CurrentEventMonState, HS_AppData_t::EMTTablePtr, HS_AppData_t::EventMonLoaded, HS_AppData_t::EventsMonitoredCount, HS_HkPacket_t::EventsMonitoredCount, HS_AppData_t::ExeCountState, CFE_ES_TaskInfo::ExecutionCounter, HS_AppData_t::HkPacket, HS_AppData, HS_BITS_PER_APPMON_ENABLE, HS_CDS_IN_USE, HS_EMT_ACT_NOACT, HS_HKREQ_RESOURCE_DBG_EID, HS_INVALID_EXECOUNT, HS_LOADED_AMT, HS_LOADED_EMT, HS_LOADED_MAT, HS_LOADED_XCT, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_APPS, HS_MAX_MONITORED_EVENTS, HS_STATE_ENABLED, HS_VerifyMsgLength(), HS_XCT_TYPE_AP_P_CHILD, HS_XCT_TYPE_APP_MAIN, HS_XCT_TYPE_DEVICE, HS_XCT_TYPE_ISR, HS_XCT_TYPE_NOTYPE, HS_HkPacket_t::InvalidEventMonCount, HS_CDSData_t::MaxResets, HS_HkPacket_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActExec, HS_HkPacket_t::MsgActExec, HS_AppData_t::MsgActsState, HS_CDSData_t::ResetsPerformed, HS_HkPacket_t::ResetsPerformed, HS_HkPacket_t::StatusFlags, HS_HkPacket_t::TlmHeader, HS_HkPacket_t::UtilCpuAvg, HS_AppData_t::UtilCpuAvg, HS_HkPacket_t::UtilCpuPeak, and HS_AppData_t::UtilCpuPeak.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.13 HS_MsgActsStatusRefresh()

```
void HS_MsgActsStatusRefresh (
    void )
```

Refresh Message Actions Status.

Description

This function gets called when HS detects that a new message actions table has been loaded: it then resets the cooldowns for all actions.

Assumptions, External Events, and Notes:

None

Definition at line 934 of file hs_cmds.c.

References HS_AppData, HS_MAX_MSG_ACT_TYPES, and HS_AppData_t::MsgActCooldown.

Referenced by HS_AcquirePointers().

12.7.2.14 HS_NoopCmd()

```
void HS_NoopCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Noop command.

Description

Processes a noop ground command.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

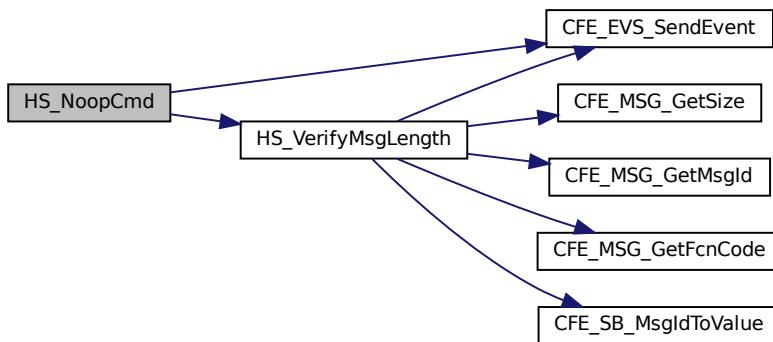
[HS_NOOP_CC](#)

Definition at line 309 of file hs_cmds.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_MAJOR_VERSION, HS_MINOR_VERSION, HS_MISSION_REV, HS_NOOP_INF_EID, HS_REVISION, HS_S.VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.15 HS_ResetCmd()

```
void HS_ResetCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Reset counters command.

Description

Processes a reset counters ground command which will reset the following HS application counters to zero:

- Command counter
- Command error counter

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

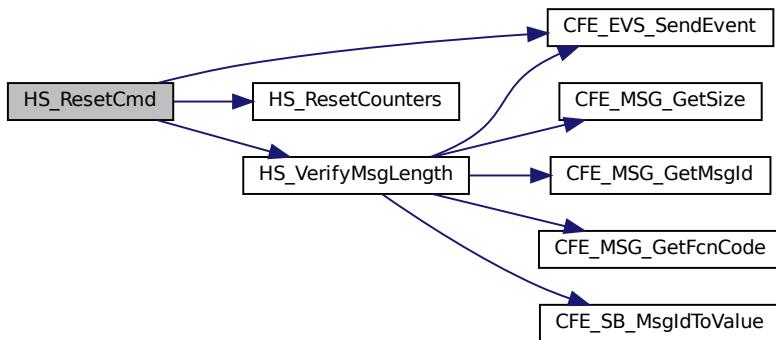
[HS_RESET_CC](#)

Definition at line 333 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_RESET_DBG_EID, HS_ResetCounters(), HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.16 HS_ResetCounters()

```
void HS_ResetCounters (
    void )
```

Reset counters.

Description

Utility function that resets housekeeping counters to zero

Assumptions, External Events, and Notes:

None

See also

[HS_ResetCmd](#)

Definition at line 356 of file hs_cmds.c.

References HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::EventsMonitoredCount, HS_AppData, and HS_AppData_t::MsgActExec.

Referenced by HS_ResetCmd().

12.7.2.17 HS_ResetResetsPerformedCmd()

```
void HS_ResetResetsPerformedCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a reset resets performed command.

Description

Resets the count of HS performed resets maintained by HS.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

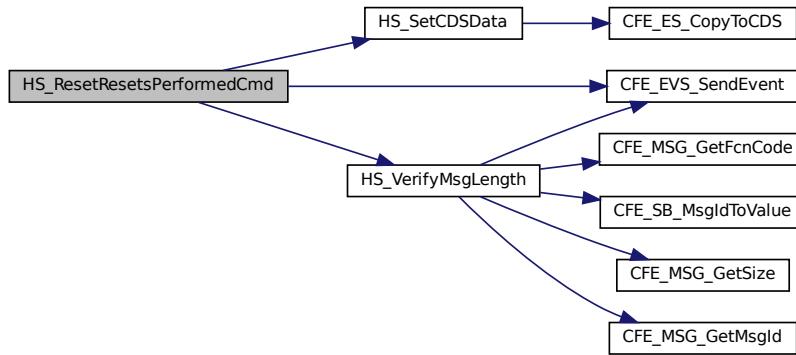
[HS_SET_MAX_RESETS_CC](#)

Definition at line 632 of file hs_cmds.c.

References HS_AppData_t::CDSData, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_RESET_RESETS_DBG_EID, HS_SetCDSData(), HS_VerifyMsgLength(), HS_CDSData_t::MaxResets, and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.7.2.18 HS_SetMaxResetsCmd()

```
void HS_SetMaxResetsCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a set max resets command.

Description

Sets the max number of HS performed resets to the specified value.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

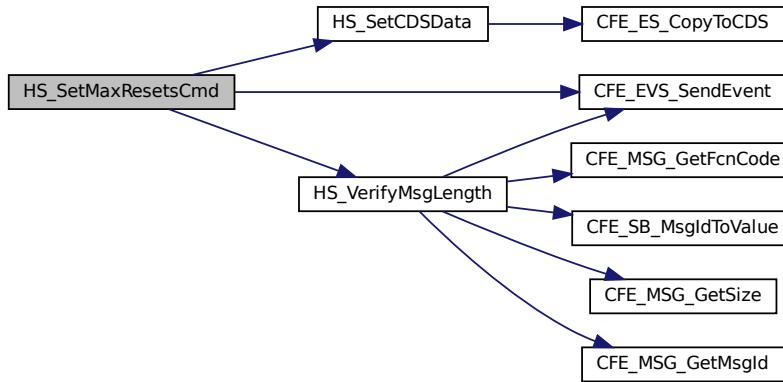
[HS_RESET_RESETS_PERFORMED_CC](#)

Definition at line 656 of file hs_cmds.c.

References HS_AppData_t::CDSData, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_SET_MAX_RESETS_DBG_EID, HS_SetCDSData(), HS_VerifyMsgLength(), HS_CDSData_t::MaxResets, HS_SetMaxResetsCmd_t::MaxResets, CFE_SB_Msg::Msg, and HS_CDSData_t::ResetsPerformed.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8 apps/hs/fsw/src/hs_cmds.h File Reference

```
#include "cfe.h"
```

Functions

- void [HS_AppPipe](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a command pipe message.
- void [HS_ResetCounters](#) (void)
Reset counters.
- void [HS_AcquirePointers](#) (void)
Manages HS tables.
- void [HS_HousekeepingReq](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Housekeeping request.
- void [HS_NoopCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)

- void [HS_ResetCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Reset counters command.
- void [HS_EnableAppMonCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process an enable critical applications monitor command.
- void [HS_DisableAppMonCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a disable critical applications monitor command.
- void [HS_EnableEventMonCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process an enable critical events monitor command.
- void [HS_DisableEventMonCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a disable critical events monitor command.
- void [HS_EnableAlivenessCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process an enable aliveness indicator command.
- void [HS_DisableAlivenessCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a disable aliveness indicator command.
- void [HS_EnableCPUHogCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process an enable CPU Hogging indicator command.
- void [HS_DisableCPUHogCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a disable CPU Hogging indicator command.
- void [HS_ResetResetsPerformedCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a reset resets performed command.
- void [HS_SetMaxResetsCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process a set max resets command.
- void [HS_AppMonStatusRefresh](#) (void)
Refresh Critical Applications Monitor Status.
- void [HS_MsgActsStatusRefresh](#) (void)
Refresh Message Actions Status.

12.8.1 Detailed Description

Specification for the CFS Health and Safety (HS) routines that handle command processing

12.8.2 Function Documentation

12.8.2.1 HS_AcquirePointers()

```
void HS_AcquirePointers (
    void )
```

Manages HS tables.

Description

Manages load requests for the AppMon, EventMon, ExeCount and MsgActs tables and update notification for the AppMon and MsgActs tables. Also releases and acquires table addresses. Gets called at the start of each processing cycle and on initialization.

Assumptions, External Events, and Notes:

None

See also

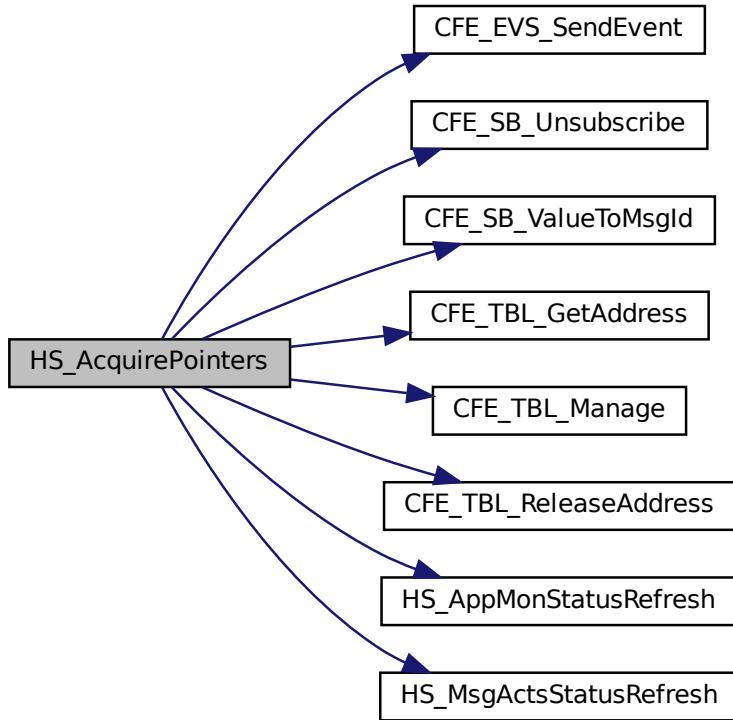
[CFE_TBL_Manage](#)

Definition at line 684 of file hs_cmds.c.

References HS_AppData_t::AMTableHandle, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonLoaded, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_Unsubscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, CFE_TBL_GetAddress(), CFE_TBL_INFO_UPDATED, CFE_TBL_Manage(), CFE_TBL_ReleaseAddress(), HS_AppData_t::CurrentAppMonState, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EMTableHandle, HS_AppData_t::EMTablePtr, HS_AppData_t::EventMonLoaded, HS_AppData_t::EventPipe, HS_AppData_t::ExeCountState, HS_AppData, HS_APPMON_GETHADDR_ERR_EID, HS_AppMonStatusRefresh(), HS_BADEMT_LONG_UNSUB_EID, HS_BADEMT_SHORT_UNSUB_EID, HS_EVENTMON_GETADDR_ERR_EID, HS_EXECOUNT_GETADDR_ERR_EID, HS_MSGACTS_GETADDR_ERR_EID, HS_MsgActsStatusRefresh(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATableHandle, HS_AppData_t::MATablePtr, and HS_AppData_t::MsgActsState.

Referenced by HS_ProcessMain(), and HS_TblInit().

Here is the call graph for this function:



12.8.2.2 HS_AppMonStatusRefresh()

```
void HS_AppMonStatusRefresh (
    void )
```

Refresh Critical Applications Monitor Status.

Description

This function gets called when HS detects that a new critical applications monitor table has been loaded or when a command to enable the critical applications monitor is received: it then refreshes the timeouts for application being monitored

Assumptions, External Events, and Notes:

None

Definition at line 892 of file hs_cmds.c.

References HS_AMTEntry_t::ActionType, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonCheckInCountdown, HS_AppData_t::AppMonEnables, HS_AppData_t::AppMonLastExeCount, CFE_SET, HS_AMTEntry_t::CycleCount, HS_AMT_ACT_NOACT, HS_AppData, HS_BITS_PER_APPMON_ENABLE, and HS_MAX_MONITORED_APPS.

Referenced by HS_AcquirePointers(), and HS_EnableAppMonCmd().

12.8.2.3 HS_AppPipe()

```
void HS_AppPipe (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a command pipe message.

Description

Processes a single software bus command pipe message. Checks the message and command IDs and calls the appropriate routine to handle the message.

Assumptions, External Events, and Notes:

None

Parameters

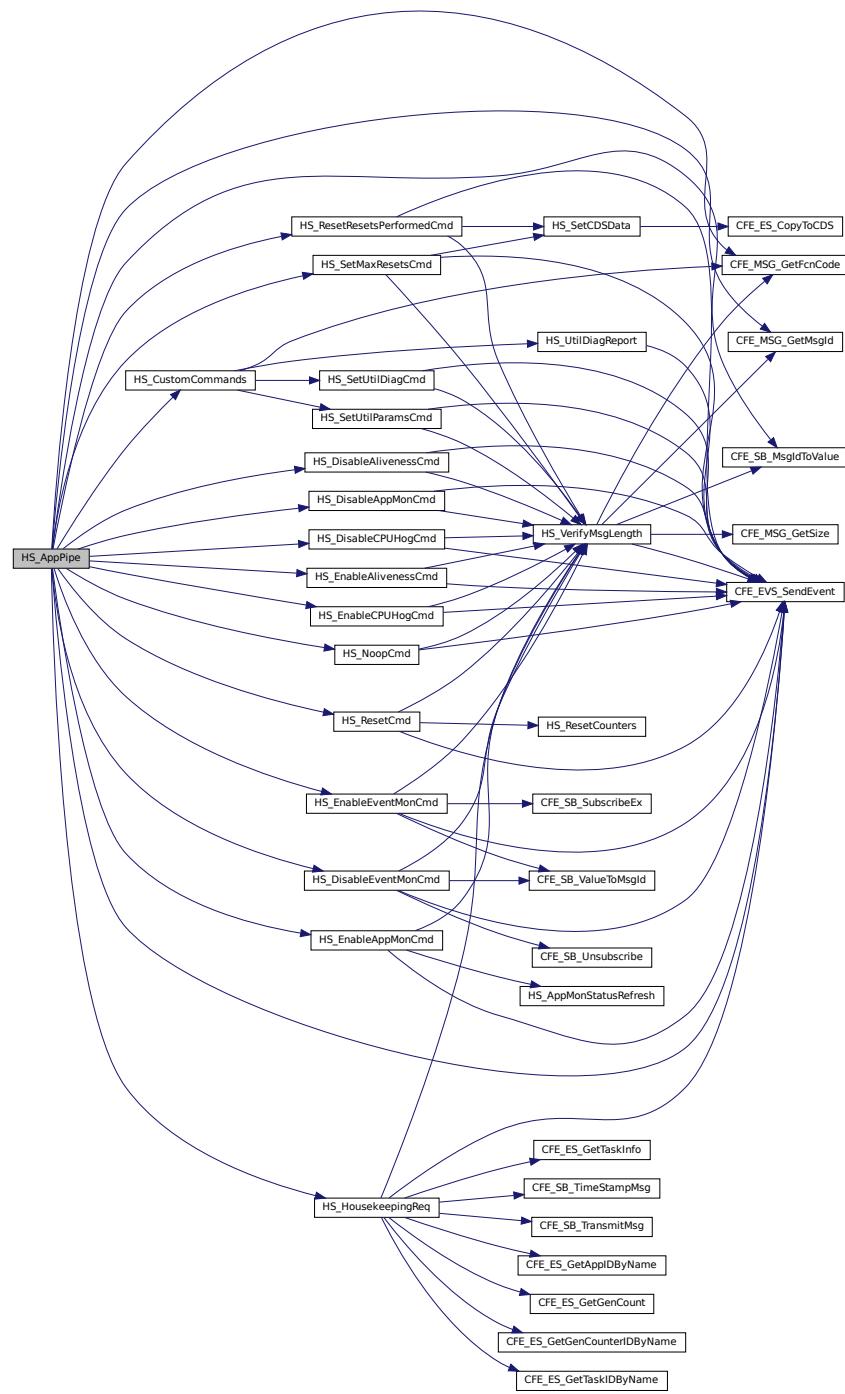
in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

Definition at line 42 of file hs_cmds.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG_GetMsgId(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), CFE_SUCCESS, HS_AppData_t::CmdErrCount, HS_AppData, HS_CC_ERR_EID, HS_CMD_MID, HS_CustomCommands(), HS_DISABLE_ALIVENESS_CC, HS_DISABLE_APPMON_CC, HS_DISABLE_CPUHOG_CC, HS_DISABLE_EVENTMON_CC, HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_ENABLE_ALIVENESS_CC, HS_ENABLE_APPMON_CC, HS_ENABLE_CPUHOG_CC, HS_ENABLE_EVENTMON_CC, HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_MID_ERR_EID, HS_NOOP_CC, HS_NoopCmd(), HS_RESET_CC, HS_RESET_RESETS_PERFORMED_CC, HS_ResetCmd(), HS_ResetResetsPerformedCmd(), HS_SEND_HK_MID, HS_SET_MAX_RESETS_CC, HS_SetMaxResetsCmd(), and CFE_SB_Msg::Msg.

Referenced by HS_ProcessCommands().

Here is the call graph for this function:



12.8.2.4 HS_DisableAlivenessCmd()

```
void HS_DisableAlivenessCmd (
```

```
const CFE_SB_Buffer_t * BufPtr )
```

Process a disable aliveness indicator command.

Description

Stops the aliveness indicator from being output on the UART.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

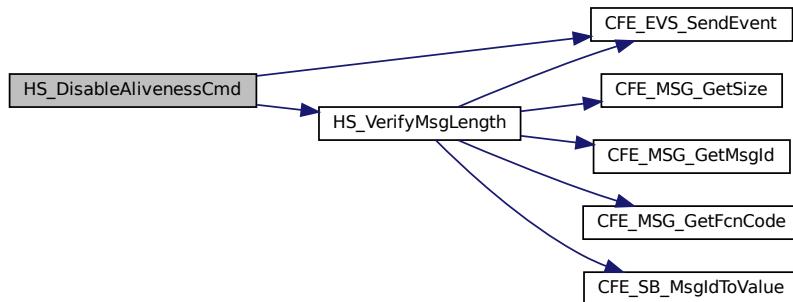
[HS_DISABLE_ALIVENESS_CC](#)

Definition at line 563 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAlivenessState, HS_AppData, HS_DISABLE_ALIVENESS_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.5 HS_DisableAppMonCmd()

```
void HS_DisableAppMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable critical applications monitor command.

Description

Stops the critical applications from be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

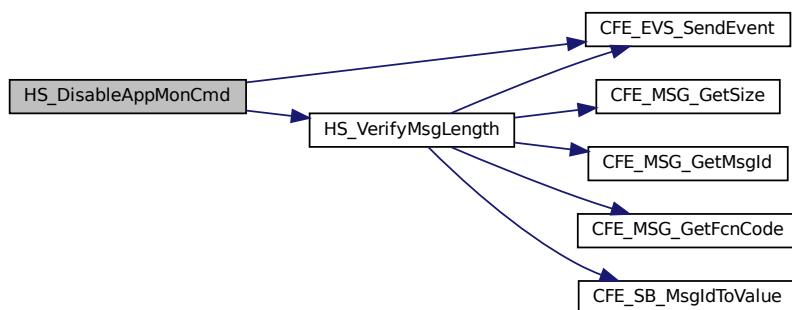
[HS_DISABLE_APPMON_CC](#)

Definition at line 396 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAppMonState, HS_AppData, HS_DISABLE_APPMON_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.6 HS_DisableCPUHogCmd()

```
void HS_DisableCPUHogCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable CPU Hogging indicator command.

Description

Stops the CPU Hogging indicator from being output as an event.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

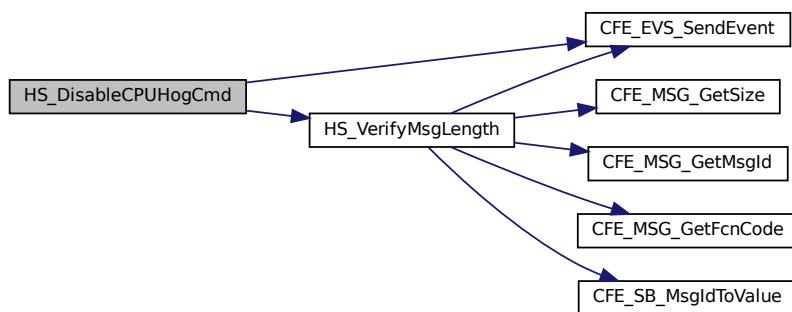
[HS_DISABLE_CPUHOG_CC](#)

Definition at line 609 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentCPUHogState, HS_AppData, HS_DISABLE_CPUHOG_DBG_EID, HS_STATE_DISABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.7 HS_DisableEventMonCmd()

```
void HS_DisableEventMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a disable critical events monitor command.

Description

Stops the critical events from be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

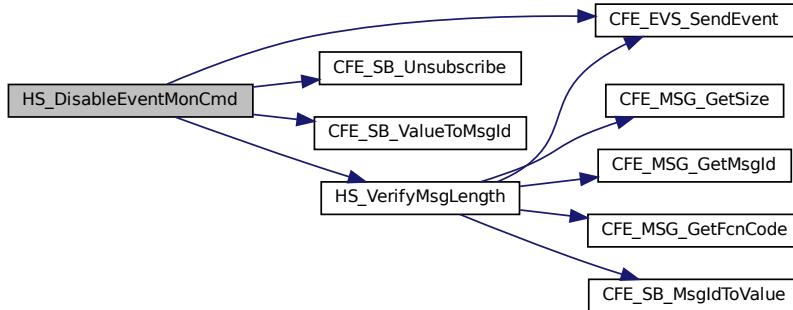
[HS_DISABLE_EVENTMON_CC](#)

Definition at line 480 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_Unsubscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData, HS_DISABLE_EVENTMON_DBG_EID, HS_EVENTMON_LONG_UNSUB_EID, HS_EVENTMON_SHORT_UNSUB_EID, HS_STATE_DISABLED, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.8 HS_EnableAlivenessCmd()

```
void HS_EnableAlivenessCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable aliveness indicator command.

Description

Allows the aliveness indicator to be output to the UART.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

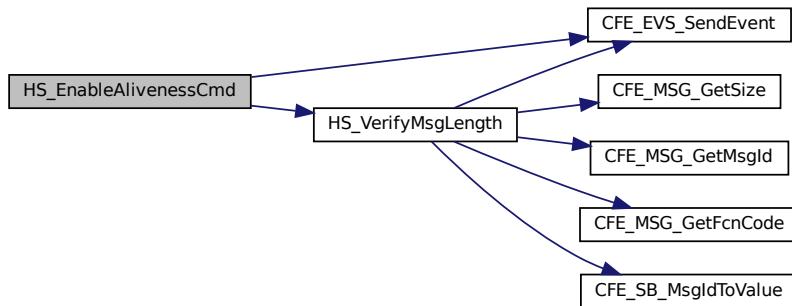
[HS_ENABLE_ALIVENESS_CC](#)

Definition at line 540 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAlivenessState, HS_AppData, HS_ENABLE_ALIVENESS_DBG_EID, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.9 HS_EnableAppMonCmd()

```
void HS_EnableAppMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable critical applications monitor command.

Description

Allows the critical applications to be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

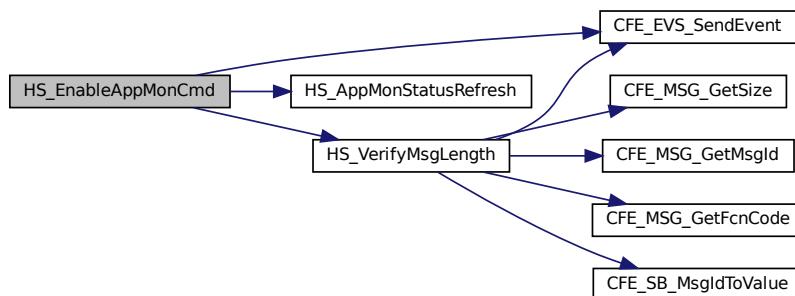
[HS_ENABLE_APPMON_CC](#)

Definition at line 372 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentAppMonState, HS_AppData, HS_AppMonStatusRefresh(), HS_ENABLE_APPMON_DBG_EID, HS_STATEENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.10 HS_EnableCPUHogCmd()

```
void HS_EnableCPUHogCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable CPU Hogging indicator command.

Description

Allows the CPU Hogging indicator to be output as an event.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

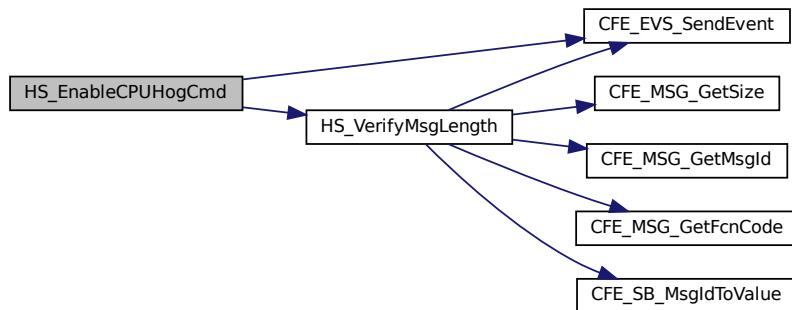
[HS_ENABLE_CPUHOG_CC](#)

Definition at line 586 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CurrentCPUHogState, HS_AppData, HS_ENABLE_CPUHOG_DBG_EID, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.11 HS_EnableEventMonCmd()

```
void HS_EnableEventMonCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process an enable critical events monitor command.

Description

Allows the critical events to be monitored.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

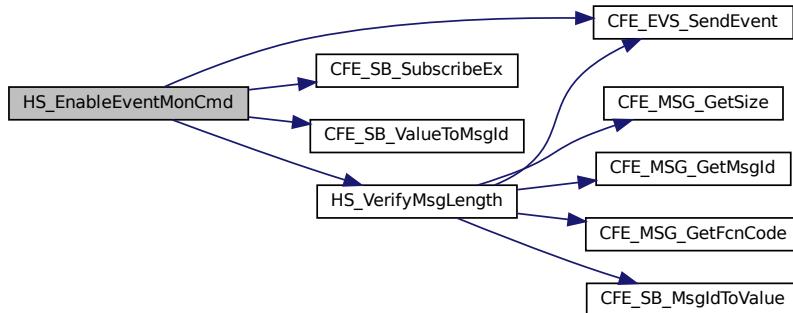
[HS_ENABLE_EVENTMON_CC](#)

Definition at line 419 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_SendEvent(), CFE_EVS_SHORT_EVENT_MSG_MID, CFE_SB_DEFAULT_QOS, CFE_SB_SubscribeEx(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentEventMonState, HS_AppData_t::EventPipe, HS_AppData, HS_ENABLE_EVENTMON_DBG_EID, HS_EVENT_PIPE_DEPTH, HS_EVENTMON_LONG_SUB_EID, HS_EVENTMON_SHORT_SUB_EID, HS_STATE_DISABLED, HS_STATE_ENABLED, HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.12 HS_HousekeepingReq()

```
void HS_HousekeepingReq (
    const CFE_SB_Buffer_t * BufPtr )
```

Housekeeping request.

Description

Processes an on-board housekeeping request message.

Assumptions, External Events, and Notes:

This message does not affect the command execution counter

Parameters

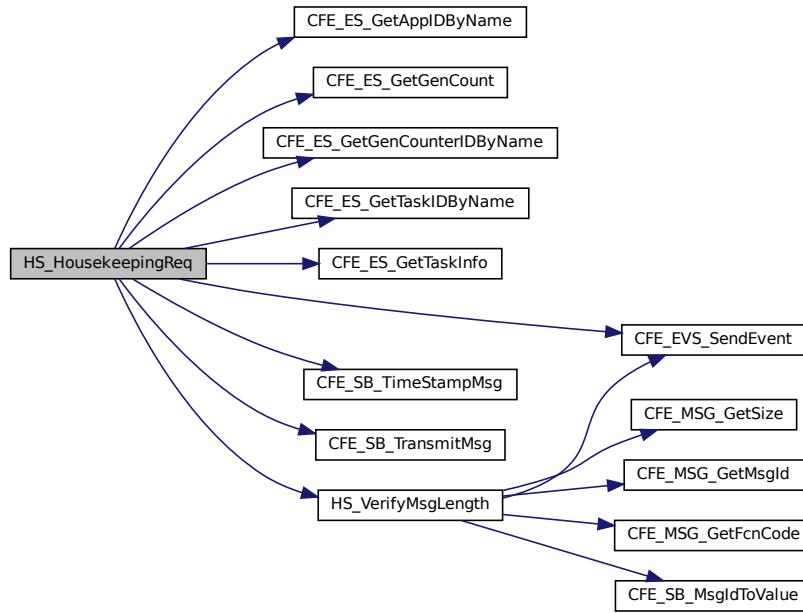
in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

Definition at line 150 of file hs_cmds.c.

References HS_EMTEEntry_t::ActionType, HS_AppData_t::AppMonEnables, HS_HkPacket_t::AppMonEnables, HS_AppData_t::AppMonLoaded, HS_EMTEEntry_t::AppName, HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_APPID_UNDEFINED, CFE_ES_COUNTERID_UNDEFINED, CFE_ES_GetAppIDByName(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CFE_ES_GetTaskIDByName(), CFE_ES_GetTaskInfo(), CFE_ES_TASKID_UNDEFINED, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_HkPacket_t::CmdCount, HS_AppData_t::CmdCount, HS_HkPacket_t::CmdErrCount, HS_AppData_t::CmdErrCount, HS_AppData_t::CurrentAlivenessState, HS_HkPacket_t::CurrentAlivenessState, HS_AppData_t::CurrentAppMonState, HS_HkPacket_t::CurrentAppMonState, HS_AppData_t::CurrentCPUHogState, HS_HkPacket_t::CurrentCPUHogState, HS_AppData_t::CurrentEventMonState, HS_HkPacket_t::CurrentEventMonState, HS_AppData_t::EMTTablePtr, HS_AppData_t::EventMonLoaded, HS_AppData_t::EventsMonitoredCount, HS_HkPacket_t::EventsMonitoredCount, HS_AppData_t::ExeCountState, CFE_ES_TaskInfo::ExecutionCounter, HS_AppData_t::HkPacket, HS_AppData, HS_BITS_PER_APPMON_ENABLE, HS_CDS_IN_USE, HS_EMT_ACT_NOACT, HS_HKREQ_RESOURCE_DBG_EID, HS_INVALID_EXECOUNT, HS_LOADED_AMT, HS_LOADED_EMT, HS_LOADED_MAT, HS_LOADED_XCT, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_APPS, HS_MAX_MONITORED_EVENTS, HS_STATE_ENABLED, HS_VerifyMsgLength(), HS_XCT_TYPE_AP_P_CHILD, HS_XCT_TYPE_APP_MAIN, HS_XCT_TYPE_DEVICE, HS_XCT_TYPE_ISR, HS_XCT_TYPE_NOTYPE, HS_HkPacket_t::InvalidEventMonCount, HS_CDSData_t::MaxResets, HS_HkPacket_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActExec, HS_HkPacket_t::MsgActExec, HS_AppData_t::MsgActsState, HS_CDSData_t::ResetsPerformed, HS_HkPacket_t::ResetsPerformed, HS_HkPacket_t::StatusFlags, HS_HkPacket_t::TlmHeader, HS_HkPacket_t::UtilCpuAvg, HS_AppData_t::UtilCpuAvg, HS_HkPacket_t::UtilCpuPeak, and HS_AppData_t::UtilCpuPeak.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.13 HS_MsgActsStatusRefresh()

```
void HS_MsgActsStatusRefresh (
    void )
```

Refresh Message Actions Status.

Description

This function gets called when HS detects that a new message actions table has been loaded: it then resets the cooldowns for all actions.

Assumptions, External Events, and Notes:

None

Definition at line 934 of file hs_cmds.c.

References HS_AppData, HS_MAX_MSG_ACT_TYPES, and HS_AppData_t::MsgActCooldown.

Referenced by HS_AcquirePointers().

12.8.2.14 HS_NoopCmd()

```
void HS_NoopCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Noop command.

Description

Processes a noop ground command.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

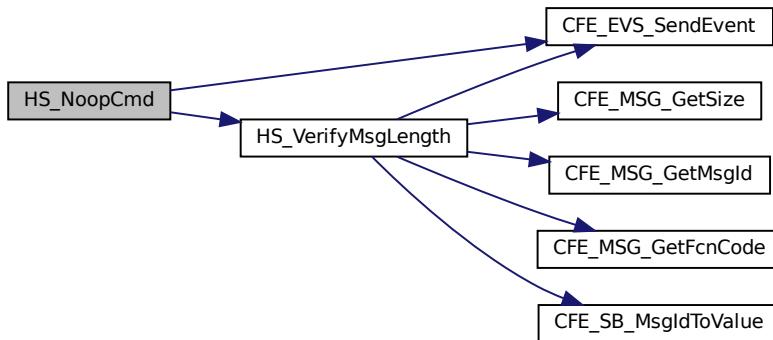
[HS_NOOP_CC](#)

Definition at line 309 of file hs_cmds.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_MAJOR_VERSION, HS_MINOR_VERSION, HS_MISSION_REV, HS_NOOP_INF_EID, HS_REVISION, HS_S.VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.15 HS_ResetCmd()

```
void HS_ResetCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Reset counters command.

Description

Processes a reset counters ground command which will reset the following HS application counters to zero:

- Command counter
- Command error counter

Assumptions, External Events, and Notes:

None

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

See also

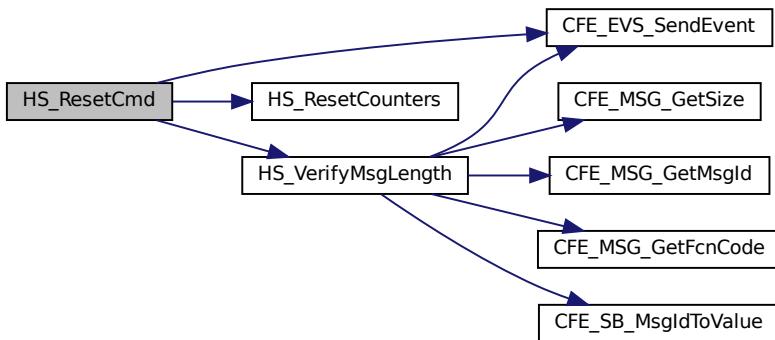
[HS_RESET_CC](#)

Definition at line 333 of file hs_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_RESET_DBG_EID, HS_ResetCounters(), HS_VerifyMsgLength(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.16 HS_ResetCounters()

```
void HS_ResetCounters (
    void )
```

Reset counters.

Description

Utility function that resets housekeeping counters to zero

Assumptions, External Events, and Notes:

None

See also

[HS_ResetCmd](#)

Definition at line 356 of file hs_cmds.c.

References HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_AppData_t::EventsMonitoredCount, HS_AppData, and HS_AppData_t::MsgActExec.

Referenced by HS_ResetCmd().

12.8.2.17 HS_ResetResetsPerformedCmd()

```
void HS_ResetResetsPerformedCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a reset resets performed command.

Description

Resets the count of HS performed resets maintained by HS.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

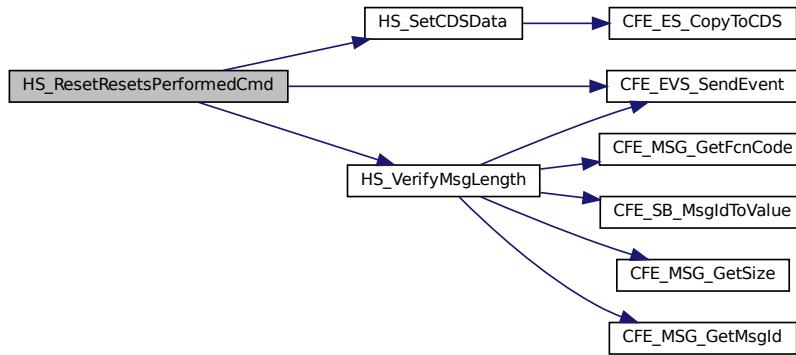
[HS_SET_MAX_RESETS_CC](#)

Definition at line 632 of file hs_cmds.c.

References HS_AppData_t::CDSData, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_RESET_RESETS_DBG_EID, HS_SetCDSData(), HS_VerifyMsgLength(), HS_CDSData_t::MaxResets, and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.8.2.18 HS_SetMaxResetsCmd()

```
void HS_SetMaxResetsCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Process a set max resets command.

Description

Sets the max number of HS performed resets to the specified value.

Assumptions, External Events, and Notes:

None

Parameters

in	BufPtr	Pointer to Software Bus buffer
----	--------	--------------------------------

See also

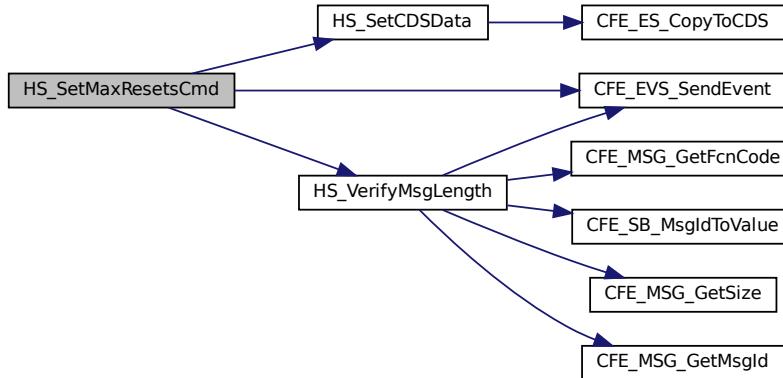
[HS_RESET_RESETS_PERFORMED_CC](#)

Definition at line 656 of file hs_cmds.c.

References HS_AppData_t::CDSData, CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_SET_MAX_RESETS_DBG_EID, HS_SetCDSData(), HS_VerifyMsgLength(), HS_CDSData_t::MaxResets, HS_SetMaxResetsCmd_t::MaxResets, CFE_SB_Msg::Msg, and HS_CDSData_t::ResetsPerformed.

Referenced by HS_AppPipe().

Here is the call graph for this function:



12.9 apps/hs/fsw/src/hs_custom.c File Reference

```

#include "cfe.h"
#include "cfe_psp.h"
#include "osapi.h"
#include "hs_app.h"
#include "hs_cmds.h"
#include "hs_msg.h"
#include "hs_utils.h"
#include "hs_custom.h"
#include "hs_events.h"
#include "hs_monitors.h"
#include "hs_perfids.h"
  
```

Functions

- void [HS_IdleTask](#) (void)
Task that increments counters.
- int32 [HS_CustomInit](#) (void)
Initialize things needed for CPU Monitoring.
- void [HS_CustomCleanup](#) (void)
Clean up the functionality used for Utilization Monitoring.
- void [HS_UtilizationIncrement](#) (void)
Increment the CPU Utilization Tracker Counter.
- void [HS_UtilizationMark](#) (void)
Mark the CPU Utilization Tracker Counter.
- void [HS_MarkIdleCallback](#) (void)
Mark Idle Time Callback from Time App.
- void [HS_CustomMonitorUtilization](#) (void)
Stub function for Utilization Monitoring.
- int32 [HS_CustomCommands](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Process Custom Commands.
- void [HS_UtilDiagReport](#) (void)
Report Utilization Diagnostics information.
- int32 [HS_CustomGetUtil](#) (void)
Stub function for Getting the Current Cycle Utilization.
- void [HS_SetUtilParamsCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Set Utilization Paramters.
- void [HS_SetUtilDiagCmd](#) (const [CFE_SB_Buffer_t](#) *BufPtr)
Set Utilization Diagnostics Paramater.

Variables

- [HS_CustomData_t](#) [HS_CustomData](#)
HS custom global.

12.9.1 Detailed Description

Health and Safety (HS) application custom component. This custom code is designed to work without modification; however the nature of specific platforms may make it desirable to modify this code to work in a more efficient way, or to provide greater functionality.

12.9.2 Function Documentation

12.9.2.1 HS_CustomCleanup()

```
void HS_CustomCleanup (
    void )
```

Clean up the functionality used for Utilization Monitoring.

Description

This function is intended to clean up everything necessary for CPU Utilization Monitoring at application termination. Currently, this terminates the Idle Task, deleting the task itself, and uncreating the 1Hz sync callback that marks the idle time. It is called at the end of [HS_AppMain](#) if HS is exiting cleanly.

Assumptions, External Events, and Notes:

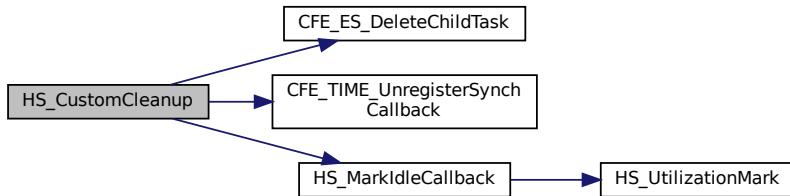
Any resources that will not be cleaned up automatically by CFE need to be cleaned up in this function.

Definition at line 142 of file hs_custom.c.

References [CFE_ES_DeleteChildTask\(\)](#), [CFE_SUCCESS](#), [CFE_TIME_UnregisterSyncCallback\(\)](#), [HS_MarkIdleCallback\(\)](#), [HS_CustomData_t::IdleTaskID](#), and [HS_CustomData_t::IdleTaskRunStatus](#).

Referenced by [HS_AppMain](#)().

Here is the call graph for this function:



12.9.2.2 HS_CustomCommands()

```
int32 HS_CustomCommands (
    const CFE_SB_Buffer_t * BufPtr )
```

Process Custom Commands.

Description

This function allows for [hs_custom.c](#) to define custom commands. It will be called for any command code not already allocated to a Health and Safety command. If a custom command is found, then it is responsible for incrementing the command processed or command error counter as appropriate.

Assumptions, External Events, and Notes:

If a command is found, this function MUST return [CFE_SUCCESS](#), otherwise it must not return [CFE_SUCCESS](#)

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

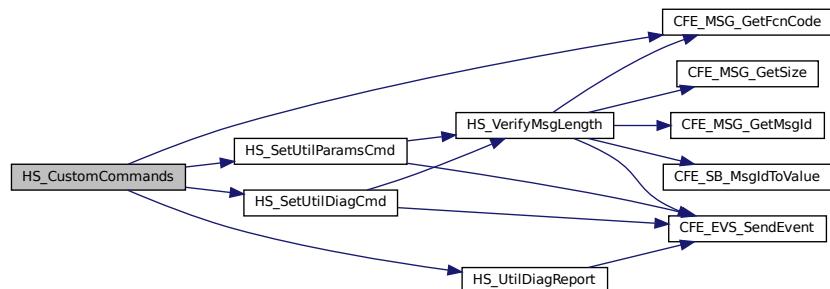
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 239 of file hs_custom.c.

References `CFE_MSG_GetFcnCode()`, `CFE_SUCCESS`, `HS_REPORT_DIAG_CC`, `HS_SET_UTIL_DIAG_CC`, `HS_SET_UTIL_PARAMS_CC`, `HS_SetUtilDiagCmd()`, `HS_SetUtilParamsCmd()`, `HS_UtilDiagReport()`, and `CFE_SB_Msg::Msg`.

Referenced by `HS_AppPipe()`.

Here is the call graph for this function:

**12.9.2.3 HS_CustomGetUtil()**

```
int32 HS_CustomGetUtil (
    void )
```

Stub function for Getting the Current Cycle Utilization.

Description

This function is used to inform the Monitor Utilization function of the current cycle utilization. It is called during [HS_MonitorUtilization](#) and should return a value between 0 and [HS_UTIL_PER_INTERVAL_TOTAL](#).

Assumptions, External Events, and Notes:

None

Returns

Current cycle utilization

Definition at line 388 of file hs_custom.c.

References HS_UTIL_PER_INTERVAL_TOTAL, HS_CustomData_t::LastIdleTaskInterval, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_MonitorUtilization().

12.9.2.4 HS_CustomInit()

```
int32 HS_CustomInit (
    void )
```

Initialize things needed for CPU Monitoring.

Description

This function is intended to set up everything necessary for CPU Utilization Monitoring at application startup. Currently, this initializes the Idle Task, spawning the task itself, and creating a 1Hz sync callback to mark the idle time. It is called at the end of [HS_AppInit](#). It may be updated to include other initializations, or modifications to already set parameters.

Assumptions, External Events, and Notes:

`CFE_SUCCESS` will be returned if all creation was performed properly.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

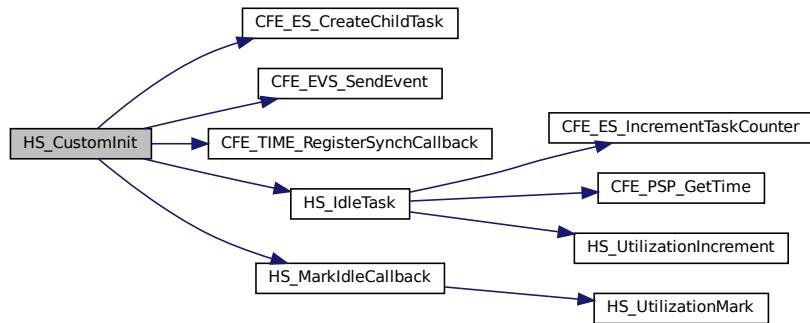
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 90 of file hs_custom.c.

References CFE_ES_CreateChildTask(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TIME_RegisterSyncCallback(), HS_CR_CHILD_TASK_ERR_EID, HS_CR_SYNC_CALLBACK_ERR_EID, HS_IDLE_TASK_FLAGS, HS_IDLE_TASK_NAME, HS_IDLE_TASK_PRIORITY, HS_IDLE_TASK_STACK_PTR, HS_IDLE_TASK_STACK_SIZE, HS_IdleTask(), HS_MarkIdleCallback(), HS_UTIL_CALLS_PER_MARK, HS_UTIL_CONV_DIV, HS_UTIL_CONV_MULT1, HS_UTIL_CONV_MULT2, HS_UTIL_DIAG_MASK, HS_UTIL_TIME_DIAG_ARRAY_MASK, HS_CustomData_t::IdleTaskID, HS_CustomData_t::IdleTaskRunStatus, HS_CustomData_t::UtilArrayMask, HS_CustomData_t::UtilCallsPerMark, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMask, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.9.2.5 HS_CustomMonitorUtilization()

```
void HS_CustomMonitorUtilization (
    void )
```

Stub function for Utilization Monitoring.

Description

This function is used as a passthrough to call [HS_MonitorUtilization](#) but can be modified to monitor utilization differently. It is called during [HS_ProcessMain](#) every HS cycle.

Assumptions, External Events, and Notes:

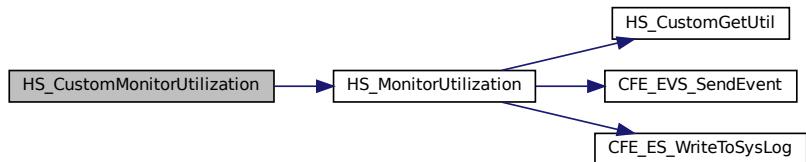
None

Definition at line 219 of file hs_custom.c.

References [HS_MonitorUtilization\(\)](#), [HS_UTIL_CYCLES_PER_INTERVAL](#), and [HS_CustomData_t::UtilCycleCounter](#).

Referenced by [HS_ProcessMain\(\)](#).

Here is the call graph for this function:



12.9.2.6 HS_IdleTask()

```
void HS_IdleTask (
    void )
```

Task that increments counters.

Description

This child task is started by the HS initialization process. It runs at the lowest priority on the system, incrementing a counter when all other tasks are idle. This counter is used to determine CPU Hogging (by being non-zero each cycle) and Utilization.

Assumptions, External Events, and Notes:

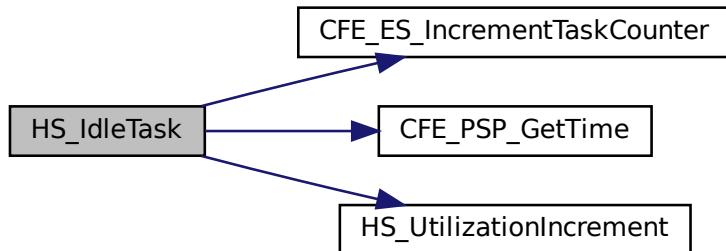
None

Definition at line 51 of file hs_custom.c.

References CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_PSP_GetTime(), CFE_SUCCESS, HS_IDLETASK_PERF_ID, HS_UtilizationIncrement(), HS_CustomData_t::IdleTaskRunStatus, HS_CustomData_t::ThisIdleTaskExec, OS_time_t::ticks, HS_CustomData_t::UtilArray, HS_CustomData_t::UtilArrayIndex, HS_CustomData_t::UtilArrayMask, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomInit().

Here is the call graph for this function:



12.9.2.7 HS_MarkIdleCallback()

```
void HS_MarkIdleCallback (
    void )
```

Mark Idle Time Callback from Time App.

Description

This function marks the idle time for the current interval.

Assumptions, External Events, and Notes:

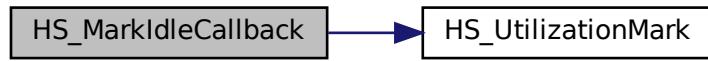
None

Definition at line 201 of file hs_custom.c.

References HS_UtilizationMark().

Referenced by HS_CustomCleanup(), and HS_CustomInit().

Here is the call graph for this function:



12.9.2.8 HS_SetUtilDiagCmd()

```
void HS_SetUtilDiagCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Set Utilization Diagnostics Parameter.

Description

This function sets the utilization diagnostics parameter.

Assumptions, External Events, and Notes:

None

Parameters

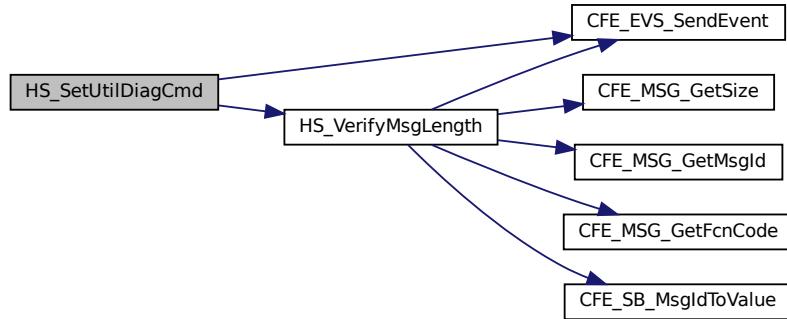
in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Definition at line 437 of file hs_custom.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_SetUtilDiag_DBG_EID, HS_VerifyMsgLength(), HS_SetUtilDiagCmd_t::Mask, CFE_SB_Msg::Msg, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomCommands().

Here is the call graph for this function:



12.9.2.9 HS_SetUtilParamsCmd()

```
void HS_SetUtilParamsCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Set Utilization Paramters.

Description

This function sets the Utilization Parameters.

Assumptions, External Events, and Notes:

None

Parameters

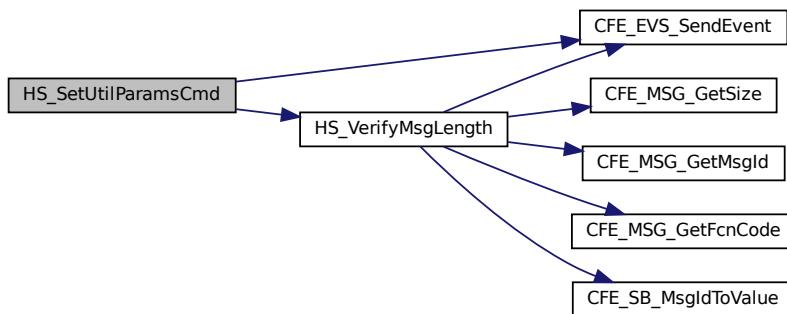
in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Definition at line 402 of file hs_custom.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_SetUtilParamsCmd_t::Div, HS_AppData, HS_SET_UTIL_PARAM_S_DBG_EID, HS_SET_UTIL_PARAMS_ERR_EID, HS_VerifyMsgLength(), CFE_SB_Msg::Msg, HS_SetUtilParamsCmd_t::Mult1, HS_SetUtilParamsCmd_t::Mult2, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_CustomCommands().

Here is the call graph for this function:



12.9.2.10 HS_UtilDiagReport()

```
void HS_UtilDiagReport (
    void )
```

Report Utilization Diagnostics information.

Description

This function reports the Utilization Diagnostics data.

Assumptions, External Events, and Notes:

None

Definition at line 276 of file hs_custom.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), HS_UTIL_DIAG_REPORT_EID, HS_UTIL_DIAG_REPORTS, HS_UTIL_TIME_DIAG_ARRAY_LENGTH, HS_CustomData_t::UtilArray, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomCommands().

Here is the call graph for this function:



12.9.2.11 HS_UtilizationIncrement()

```
void HS_UtilizationIncrement (
    void )
```

Increment the CPU Utilization Tracker Counter.

Description

Utility function that increments the CPU Utilization tracking counter, called by Idle Task. This counter is used to determine both utilization and CPU Hogging.

Assumptions, External Events, and Notes:

None

See also

[HS_UtilizationMark](#)

Definition at line 166 of file hs_custom.c.

References HS_CustomData_t::ThisIdleTaskExec.

Referenced by HS_IdleTask().

12.9.2.12 HS_UtilizationMark()

```
void HS_UtilizationMark (
    void )
```

Mark the CPU Utilization Tracker Counter.

Description

Utility function that marks the CPU Utilization tracking counter while saving the previous value to a variable for use in calculating CPU Utilization and hogging.

Assumptions, External Events, and Notes:

None

See also

[HS_UtilizationIncrement](#)

Definition at line 179 of file hs_custom.c.

References HS_CustomData_t::LastIdleTaskExec, HS_CustomData_t::LastIdleTaskInterval, HS_CustomData_t::ThisIdleTaskExec, and HS_CustomData_t::UtilCallsPerMark.

Referenced by HS_MarkIdleCallback().

12.9.3 Variable Documentation

12.9.3.1 HS_CustomData

[HS_CustomData_t](#) HS_CustomData

HS custom global.

Definition at line 43 of file hs_custom.c.

12.10 apps/fsf/src/hs_custom.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct [HS_SetUtilParamsCmd_t](#)
Set Utililiztion Parameters Command.
- struct [HS_SetUtilDiagCmd_t](#)
Set Utilization Diagnostics Command.
- struct [HS_CustomData_t](#)
HS custom global structure.

Macros

- #define [HS_UTIL_DIAG_REPORTS](#) 4

- #define [HS_REPORT_DIAG_CC](#) 12
Report Util Diagnostics.
- #define [HS_SET_UTIL_PARAMS_CC](#) 13
Set Utilization Calibration Parameters.
- #define [HS_SET_UTIL_DIAG_CC](#) 14
Set Utilization Diagnostics Parameter.

- #define [HS_CR_CHILD_TASK_ERR_EID](#) 101
HS CPU Utilization Monitoring Create Child Task Failed Event ID.
- #define [HS_CR_SYNC_CALLBACK_ERR_EID](#) 102
HS CPU Utilization Monitoring Register Sync Callback Failed Event ID.
- #define [HS_UTIL_DIAG_REPORT_EID](#) 103
HS Report Diagnostics Command Event ID.
- #define [HS_SET_UTIL_PARAMS_DBG_EID](#) 104
HS Set Utilization Paramaters Command Event ID.
- #define [HS_SET_UTIL_PARAMS_ERR_EID](#) 105
HS Set Utilization Parameters Zero Value Event ID.
- #define [HS_SET_UTIL_DIAG_DBG_EID](#) 106
HS Set Utilization Diagnostics Command Event ID.

Functions

- **int32 HS_CustomInit (void)**
Initialize things needed for CPU Monitoring.
- **void HS_CustomCleanup (void)**
Clean up the functionality used for Utilization Monitoring.
- **void HS_CustomMonitorUtilization (void)**
Stub function for Utilization Monitoring.
- **int32 HS_CustomGetUtil (void)**
Stub function for Getting the Current Cycle Utilization.
- **int32 HS_CustomCommands (const CFE_SB_Buffer_t *BufPtr)**
Process Custom Commands.
- **void HS_IdleTask (void)**
Task that increments counters.
- **void HS_UtilizationIncrement (void)**
Increment the CPU Utilization Tracker Counter.
- **void HS_UtilizationMark (void)**
Mark the CPU Utilization Tracker Counter.
- **void HS_MarkIdleCallback (void)**
Mark Idle Time Callback from Time App.
- **void HS_UtilDiagReport (void)**
Report Utilization Diagnostics information.
- **void HS_SetUtilParamsCmd (const CFE_SB_Buffer_t *BufPtr)**
Set Utilization Paramters.
- **void HS_SetUtilDiagCmd (const CFE_SB_Buffer_t *BufPtr)**
Set Utilization Diagnostics Paramater.

Variables

- **HS_CustomData_t HS_CustomData**
HS custom global.

12.10.1 Detailed Description

Specification for the CFS Health and Safety (HS) mission specific custom function interface

12.10.2 Macro Definition Documentation

12.10.2.1 HS_UTIL_DIAG_REPORTS

```
#define HS_UTIL_DIAG_REPORTS 4
```

Definition at line 36 of file hs_custom.h.

Referenced by HS_UtilDiagReport().

12.10.3 Function Documentation

12.10.3.1 HS_CustomCleanup()

```
void HS_CustomCleanup (
    void )
```

Clean up the functionality used for Utilization Monitoring.

Description

This function is intended to clean up everything necessary for CPU Utilization Monitoring at application termination. Currently, this terminates the Idle Task, deleting the task itself, and uncreating the 1Hz sync callback that marks the idle time. It is called at the end of [HS_AppMain](#) if HS is exiting cleanly.

Assumptions, External Events, and Notes:

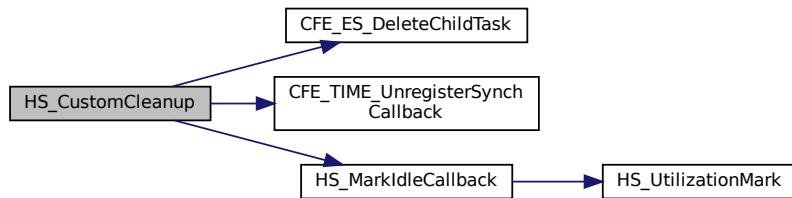
Any resources that will not be cleaned up automatically by CFE need to be cleaned up in this function.

Definition at line 142 of file hs_custom.c.

References [CFE_ES_DeleteChildTask\(\)](#), [CFE_SUCCESS](#), [CFE_TIME_UnregisterSyncCallback\(\)](#), [HS_MarkIdleCallback\(\)](#), [HS_CustomData_t::IdleTaskID](#), and [HS_CustomData_t::IdleTaskRunStatus](#).

Referenced by [HS_AppMain\(\)](#).

Here is the call graph for this function:



12.10.3.2 HS_CustomCommands()

```
int32 HS_CustomCommands (
    const CFE_SB_Buffer_t * BufPtr )
```

Process Custom Commands.

Description

This function allows for [hs_custom.c](#) to define custom commands. It will be called for any command code not already allocated to a Health and Safety command. If a custom command is found, then it is responsible for incrementing the command processed or command error counter as appropriate.

Assumptions, External Events, and Notes:

If a command is found, this function MUST return [CFE_SUCCESS](#), otherwise it must not return [CFE_SUCCESS](#)

Parameters

in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

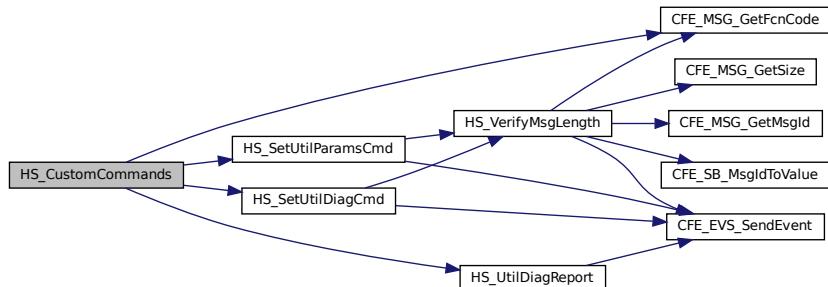
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 239 of file hs_custom.c.

References CFE_MSG_GetFcnCode(), CFE_SUCCESS, HS_REPORT_DIAG_CC, HS_SET_UTIL_DIAG_CC, HS_SET_UTIL_PARAMS_CC, HS_SetUtilDiagCmd(), HS_SetUtilParamsCmd(), HS_UtilDiagReport(), and CFE_SB_Msg::Msg.

Referenced by HS_AppPipe().

Here is the call graph for this function:

**12.10.3.3 HS_CustomGetUtil()**

```
int32 HS_CustomGetUtil (
    void )
```

Stub function for Getting the Current Cycle Utilization.

Description

This function is used to inform the Monitor Utilization function of the current cycle utilization. It is called during [HS_MonitorUtilization](#) and should return a value between 0 and [HS_UTIL_PER_INTERVAL_TOTAL](#).

Assumptions, External Events, and Notes:

None

Returns

Current cycle utilization

Definition at line 388 of file hs_custom.c.

References HS_UTIL_PER_INTERVAL_TOTAL, HS_CustomData_t::LastIdleTaskInterval, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_MonitorUtilization().

12.10.3.4 HS_CustomInit()

```
int32 HS_CustomInit (
    void )
```

Initialize things needed for CPU Monitoring.

Description

This function is intended to set up everything necessary for CPU Utilization Monitoring at application startup. Currently, this initializes the Idle Task, spawning the task itself, and creating a 1Hz sync callback to mark the idle time. It is called at the end of [HS_AppInit](#). It may be updated to include other initializations, or modifications to already set parameters.

Assumptions, External Events, and Notes:

`CFE_SUCCESS` will be returned if all creation was performed properly.

Returns

Execution status, see [cFE Return Code Defines](#)

Return values

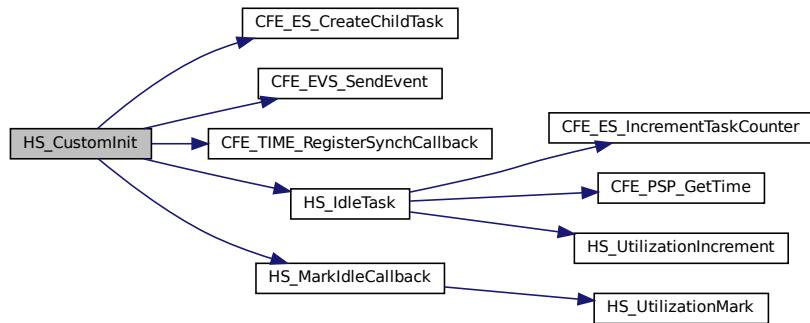
CFE_SUCCESS	Successful execution.
-----------------------------	-----------------------

Definition at line 90 of file hs_custom.c.

References CFE_ES_CreateChildTask(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_TIME_RegisterSyncCallback(), HS_CR_CHILD_TASK_ERR_EID, HS_CR_SYNC_CALLBACK_ERR_EID, HS_IDLE_TASK_FLAGS, HS_IDLE_TASK_NAME, HS_IDLE_TASK_PRIORITY, HS_IDLE_TASK_STACK_PTR, HS_IDLE_TASK_STACK_SIZE, HS_IdleTask(), HS_MarkIdleCallback(), HS_UTIL_CALLS_PER_MARK, HS_UTIL_CONV_DIV, HS_UTIL_CONV_MULT1, HS_UTIL_CONV_MULT2, HS_UTIL_DIAG_MASK, HS_UTIL_TIME_DIAG_ARRAY_MASK, HS_CustomData_t::IdleTaskID, HS_CustomData_t::IdleTaskRunStatus, HS_CustomData_t::UtilArrayMask, HS_CustomData_t::UtilCallsPerMark, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMask, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_AppInit().

Here is the call graph for this function:



12.10.3.5 HS_CustomMonitorUtilization()

```
void HS_CustomMonitorUtilization (
    void )
```

Stub function for Utilization Monitoring.

Description

This function is used as a passthrough to call [HS_MonitorUtilization](#) but can be modified to monitor utilization differently. It is called during [HS_ProcessMain](#) every HS cycle.

Assumptions, External Events, and Notes:

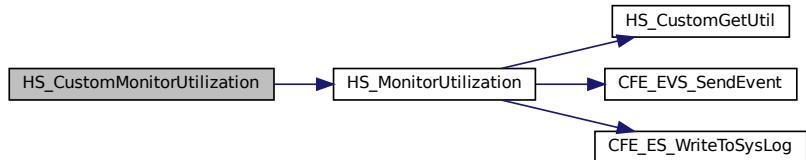
None

Definition at line 219 of file hs_custom.c.

References [HS_MonitorUtilization\(\)](#), [HS_UTIL_CYCLES_PER_INTERVAL](#), and [HS_CustomData_t::UtilCycleCounter](#).

Referenced by [HS_ProcessMain\(\)](#).

Here is the call graph for this function:



12.10.3.6 HS_IdleTask()

```
void HS_IdleTask (
    void )
```

Task that increments counters.

Description

This child task is started by the HS initialization process. It runs at the lowest priority on the system, incrementing a counter when all other tasks are idle. This counter is used to determine CPU Hogging (by being non-zero each cycle) and Utilization.

Assumptions, External Events, and Notes:

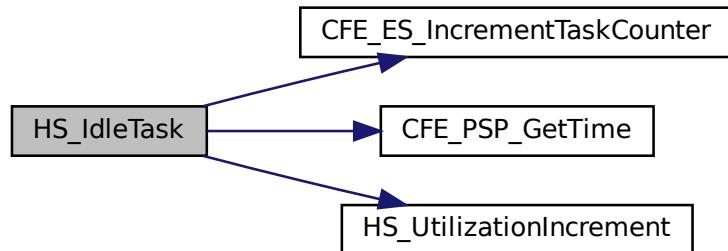
None

Definition at line 51 of file hs_custom.c.

References CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_PSP_GetTime(), CFE_SUCCESS, HS_IDLETASK_PERF_ID, HS_UtilizationIncrement(), HS_CustomData_t::IdleTaskRunStatus, HS_CustomData_t::ThisIdleTaskExec, OS_time_t::ticks, HS_CustomData_t::UtilArray, HS_CustomData_t::UtilArrayIndex, HS_CustomData_t::UtilArrayMask, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomInit().

Here is the call graph for this function:



12.10.3.7 HS_MarkIdleCallback()

```
void HS_MarkIdleCallback (
    void )
```

Mark Idle Time Callback from Time App.

Description

This function marks the idle time for the current interval.

Assumptions, External Events, and Notes:

None

Definition at line 201 of file hs_custom.c.

References HS_UtilizationMark().

Referenced by HS_CustomCleanup(), and HS_CustomInit().

Here is the call graph for this function:



12.10.3.8 HS_SetUtilDiagCmd()

```
void HS_SetUtilDiagCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Set Utilization Diagnostics Parameter.

Description

This function sets the utilization diagnostics parameter.

Assumptions, External Events, and Notes:

None

Parameters

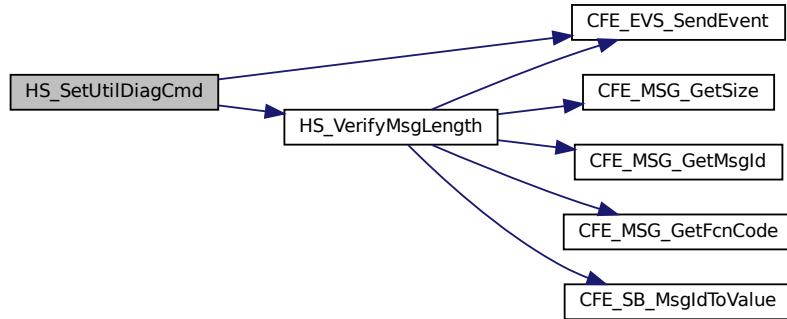
in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Definition at line 437 of file hs_custom.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData, HS_SetUtilDiag_DBG_EID, HS_VerifyMsgLength(), HS_SetUtilDiagCmd_t::Mask, CFE_SB_Msg::Msg, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomCommands().

Here is the call graph for this function:

**12.10.3.9 HS_SetUtilParamsCmd()**

```
void HS_SetUtilParamsCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Set Utilization Paramters.

Description

This function sets the Utilization Parameters.

Assumptions, External Events, and Notes:

None

Parameters

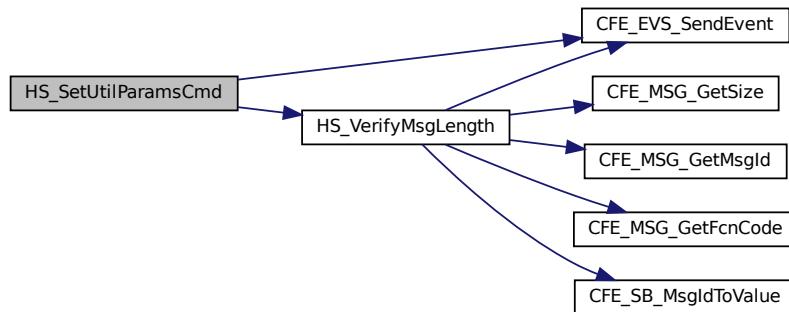
in	<i>BufPtr</i>	Pointer to Software Bus buffer
----	---------------	--------------------------------

Definition at line 402 of file hs_custom.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), HS_AppData_t::CmdCount, HS_AppData_t::CmdErrCount, HS_SetUtilParamsCmd_t::Div, HS_AppData, HS_SET_UTIL_PARAM_S_DBG_EID, HS_SET_UTIL_PARAMS_ERR_EID, HS_VerifyMsgLength(), CFE_SB_Msg::Msg, HS_SetUtilParamsCmd_t::Mult1, HS_SetUtilParamsCmd_t::Mult2, HS_CustomData_t::UtilDiv, HS_CustomData_t::UtilMult1, and HS_CustomData_t::UtilMult2.

Referenced by HS_CustomCommands().

Here is the call graph for this function:



12.10.3.10 HS_UtilDiagReport()

```
void HS_UtilDiagReport (
    void )
```

Report Utilization Diagnostics information.

Description

This function reports the Utilization Diagnostics data.

Assumptions, External Events, and Notes:

None

Definition at line 276 of file hs_custom.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), HS_UTIL_DIAG_REPORT_EID, HS_UTIL_DIAG_REPORTS, HS_UTIL_TIME_DIAG_ARRAY_LENGTH, HS_CustomData_t::UtilArray, and HS_CustomData_t::UtilMask.

Referenced by HS_CustomCommands().

Here is the call graph for this function:



12.10.3.11 HS_UtilizationIncrement()

```
void HS_UtilizationIncrement (
    void )
```

Increment the CPU Utilization Tracker Counter.

Description

Utility function that increments the CPU Utilization tracking counter, called by Idle Task. This counter is used to determine both utilization and CPU Hogging.

Assumptions, External Events, and Notes:

None

See also

[HS_UtilizationMark](#)

Definition at line 166 of file hs_custom.c.

References HS_CustomData_t::ThisIdleTaskExec.

Referenced by HS_IdleTask().

12.10.3.12 HS_UtilizationMark()

```
void HS_UtilizationMark (
    void )
```

Mark the CPU Utilization Tracker Counter.

Description

Utility function that marks the CPU Utilization tracking counter while saving the previous value to a variable for use in calculating CPU Utilization and hogging.

Assumptions, External Events, and Notes:

None

See also

[HS_UtilizationIncrement](#)

Definition at line 179 of file hs_custom.c.

References HS_CustomData_t::LastIdleTaskExec, HS_CustomData_t::LastIdleTaskInterval, HS_CustomData_t::ThisIdleTaskExec, and HS_CustomData_t::UtilCallsPerMark.

Referenced by HS_MarkIdleCallback().

12.10.4 Variable Documentation

12.10.4.1 HS_CustomData

[HS_CustomData_t](#) HS_CustomData

HS custom global.

Definition at line 43 of file hs_custom.c.

12.11 apps/hs/fsw/src/hs_events.h File Reference

Macros

- #define HS_INIT_EID 1
HS Initialization Event ID.
- #define HS_APP_EXIT_EID 2
HS Application Fatal Termination Event ID.
- #define HS_CDS_RESTORE_ERR_EID 3
HS CDS Restore Failed Event ID.
- #define HS_CR_CMD_PIPE_ERR_EID 4
HS Create Software Bus Command Pipe Failed Event ID.
- #define HS_CR_EVENT_PIPE_ERR_EID 5
HS Create Software Bus Event Pipe Failed Event ID.
- #define HS_CR_WAKEUP_PIPE_ERR_EID 6
HS Create Software Bus Wakeup Pipe Failed Event ID.
- #define HS_SUB_LONG_EVS_ERR_EID 7
HS Initialization Subscribe To Long Events Failed Event ID.
- #define HS_SUB_REQ_ERR_EID 8
HS Subscribe To Housekeeping Request Failed Event ID.
- #define HS_SUB_CMD_ERR_EID 9
HS Subscribe To Ground Commands Failed Event ID.
- #define HS_AMT_REG_ERR_EID 10
HS AppMon Table Register Failed Event ID.
- #define HS_EMT_REG_ERR_EID 11
HS EventMon Table Register Failed Event ID.
- #define HS_XCT_REG_ERR_EID 12
HS ExeCount Table Register Failed Event ID.
- #define HS_MAT_REG_ERR_EID 13
HS MsgActs Table Register Failed Event ID.
- #define HS_AMT_LD_ERR_EID 14
HS AppMon Table Load Failed Event ID.
- #define HS_EMT_LD_ERR_EID 15
HS EventMon Table Load Failed Event ID.
- #define HS_XCT_LD_ERR_EID 16
HS ExeCount Table Load Failed Event ID.
- #define HS_MAT_LD_ERR_EID 17
HS MsgActs Table Load Failed Event ID.
- #define HS_CDS_CORRUPT_ERR_EID 18
HS CDS Data Corrupt Resetting Data Event ID.
- #define HS_CC_ERR_EID 19
HS Command Code Invalid Event ID.
- #define HS_MID_ERR_EID 20
HS Command Pipe Message ID Invalid Event ID.
- #define HS_HKREQ_LEN_ERR_EID 21
HS Housekeeping Request Message Length Invalid Event ID.
- #define HS_LEN_ERR_EID 22

- `#define HS_NOOP_INF_EID 23`
HS Ground Command Length Invalid Event ID.
- `#define HS_RESET_DBG_EID 24`
HS No-op Command Event ID.
- `#define HS_ENABLE_APPMON_DBG_EID 25`
HS Reset Counters Command Event ID.
- `#define HS_DISABLE_APPMON_DBG_EID 26`
HS Enable Application Monitoring Command Event ID.
- `#define HS_ENABLE_EVENTMON_DBG_EID 27`
HS Disable Application Monitoring Command Event ID.
- `#define HS_DISABLE_EVENTMON_DBG_EID 28`
HS Enable Event Monitoring Command Event ID.
- `#define HS_ENABLE_ALIVENESS_DBG_EID 29`
HS Disable Event Monitoring Command Event ID.
- `#define HS_DISABLE_ALIVENESS_DBG_EID 30`
HS Enable Aliveness Indicator Command Event ID.
- `#define HS_RESET_RESETS_DBG_EID 31`
HS Disable Aliveness Indicator Command Event ID.
- `#define HS_SET_MAX_RESETS_DBG_EID 32`
HS Reset Process Reset Counter Command Event ID.
- `#define HS_APPMON_GETADDR_ERR_EID 33`
HS Set Max Resets Command Event ID.
- `#define HS_EVENTMON_GETADDR_ERR_EID 34`
HS AppMon Table Get Address Failed Event ID.
- `#define HS_EXECOUNT_GETADDR_ERR_EID 35`
HS EventMon Table Get Address Failed Event ID.
- `#define HS_MSGACTS_GETADDR_ERR_EID 36`
HS ExeCount Table Get Address Failed Event ID.
- `#define HS_RESET_LIMIT_ERR_EID 37`
HS MsgActs Table Get Address Failed Event ID.
- `#define HS_APPMON_APPNAME_ERR_EID 38`
HS Processor Reset Action Limit Reached No Reset Performed Event ID.
- `#define HS_APPMON_RESTART_ERR_EID 39`
HS App Monitor App Name Not Found Event ID.
- `#define HS_APPMON_NOT_RESTARTED_ERR_EID 40`
HS App Monitor Application Restart Event ID.
- `#define HS_APPMON_FAIL_ERR_EID 41`
HS App Monitor Application Restart Failed Event ID.
- `#define HS_APPMON_PROC_ERR_EID 42`
HS App Monitor Event Only Event ID.
- `#define HS_APPMON_MSGACTS_ERR_EID 43`
HS App Monitor Processor Reset Event ID.
- `#define HS_EVENTMON_MSGACTS_ERR_EID 44`
HS App Monitor Message Action Event ID.
- `#define HS_EVENTMON_PROC_ERR_EID 45`
HS Event Monitor Message Action Event ID.
- `#define HS_EVENTMON_RESET_ERR_EID 46`
HS Event Monitor Processor Reset Event ID.

- #define HS_EVENTMON_RESTART_ERR_EID 46
HS Event Monitor Application Restart Event ID.
- #define HS_EVENTMON_NOT_RESTARTED_ERR_EID 47
HS Event Monitor Application Restart Failed Event ID.
- #define HS_EVENTMON_DELETE_ERR_EID 48
HS Event Monitor Application Delete Event ID.
- #define HS_EVENTMON_NOT_DELETED_ERR_EID 49
HS Event Monitor Application Delete Failed Event ID.
- #define HS_AMTVAL_INF_EID 50
HS AppMon Table Verification Results Event ID.
- #define HS_AMTVAL_ERR_EID 51
HS AppMon Table Verification Failed Results Event ID.
- #define HS_EMTVAL_INF_EID 52
HS EventMon Table Verification Results Event ID.
- #define HS_EMTVAL_ERR_EID 53
HS EventMon Table Verification Failed Event ID.
- #define HS_XCTVAL_INF_EID 54
HS ExeCount Table Verification Results Event ID.
- #define HS_XCTVAL_ERR_EID 55
HS ExeCount Table Verification Failed Event ID.
- #define HS_MATVAL_INF_EID 56
HS MsgActs Table Verification Results Event ID.
- #define HS_MATVAL_ERR_EID 57
HS MsgActs Table Verification Failed Event ID.
- #define HS_DISABLE_APPMON_ERR_EID 58
HS Application Monitoring Disabled Due To Table Load Failure Event ID.
- #define HS_DISABLE_EVENTMON_ERR_EID 59
HS Event Monitoring Disabled Due To Table Load Failure Event ID.
- #define HS_SUB_WAKEUP_ERR_EID 60
HS Subscribe To Wakeup Message Failed Event ID.
- #define HS_CPUMON_HOGGING_ERR_EID 61
HS CPU Hogging Detected Event ID.
- #define HS_ENABLE_CPUHOG_DBG_EID 64
HS CPU Hogging Detection Enabled Event ID.
- #define HS_DISABLE_CPUHOG_DBG_EID 65
HS CPU Hogging Detection Disabled Event ID.
- #define HS_EVENTMON_LONG_SUB_EID 66
HS Event Monitor Enable Subscribe To Long Events Failed Event ID.
- #define HS_EVENTMON_SHORT_SUB_EID 67
HS Event Monitor Enable Subscribe to Short Events Failed Event ID.
- #define HS_EVENTMON_LONG_UNSUB_EID 68
HS Event Monitor Disable Unsubscribe From Long Events Failed Event ID.
- #define HS_EVENTMON_SHORT_UNSUB_EID 69
HS Event Monitor Disable Unsubscribe From Short Events Failed Event ID.
- #define HS_BADEMT_LONG_UNSUB_EID 70
HS EventMon Table Bad Unsubscribing From Long Events Failed Event ID.
- #define HS_BADEMT_SHORT_UNSUB_EID 71

- `#define HS_APPMON_APPNAME_DBG_EID 72`
HS EventMon Table Bad Unsubscribing From Short Events Failed Event ID.
- `#define HS_HKREQ_RESOURCE_DBG_EID 73`
HS App Monitor App Name Not Found Event ID.
- `#define HS_CUSTOM_INIT_ERR_EID 74`
HS Housekeeping Request Unknown Resource Event ID.
- `#define HS_AM_TBL_NULL_ERR_EID 75`
HS Custom Initialization Failed Event ID.
- `#define HS_EM_TBL_NULL_ERR_EID 76`
HS AppMon Table Validation Null Pointer Event ID.
- `#define HS_XC_TBL_NULL_ERR_EID 77`
HS ExeCount Table Validation Null Pointer Event ID.
- `#define HS_MA_TBL_NULL_ERR_EID 78`
HS MsgActs Table Validation Null Pointer Event ID.
- `#define HS_SUB_SHORT_EVTS_ERR_EID 79`
HS Initialization Subscribe To Short Events Failed Event ID.

12.11.1 Detailed Description

Specification for the CFS Health and Safety (HS) event identifiers.

12.12 apps/hs/fsw/src/hs_monitors.c File Reference

```
#include "hs_app.h"
#include "hs_monitors.h"
#include "hs_custom.h"
#include "hs_tbldefs.h"
#include "hs_events.h"
#include "hs_utils.h"
#include "cfe_evs_msg.h"
#include <string.h>
```

Functions

- `void HS_MonitorApplications (void)`
Check execution status of each app in AppMon table.
- `void HS_MonitorEvent (const CFE_EVS_LongEventTlm_t *EventPtr)`
Search the EventMon table for matches to the incoming event.
- `void HS_MonitorUtilization (void)`
Monitor the utilization tracker counter.
- `int32 HS_ValidateAMTable (void *TableData)`
Validate application monitor table.
- `int32 HS_ValidateEMTable (void *TableData)`
Validate event monitor table.
- `int32 HS_ValidateMATable (void *TableData)`
Validate message actions table.
- `void HS_SetCDSData (uint16 ResetsPerformed, uint16 MaxResets)`
Update and store CDS data.

12.12.1 Detailed Description

Functions used for CFS Health and Safety Monitors for Applications and Events

12.12.2 Function Documentation

12.12.2.1 HS_MonitorApplications()

```
void HS_MonitorApplications (
    void )
```

Check execution status of each app in AppMon table.

Description

Cycles through the Application Monitor Table checking the current execution count for each monitored application. If the count fails to increment for the table specified duration, the table specified action is taken.

Assumptions, External Events, and Notes:

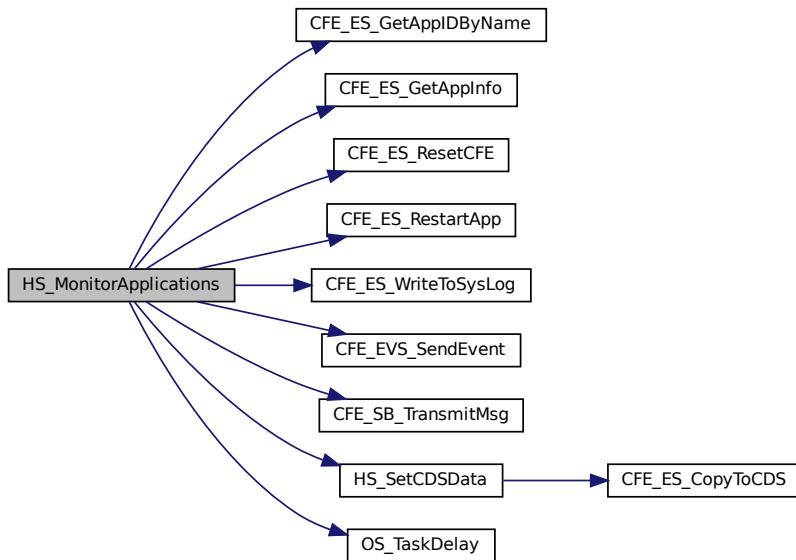
None

Definition at line 44 of file hs_monitors.c.

References HS_AMTEntry_t::ActionType, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonCheckInCountdown, HS_AppData_t::AppMonEnables, HS_AppData_t::AppMonLastExeCount, HS_AMTEntry_t::AppName, HS_AppData_t::CDSData, CFE_CLR, CFE_ES_APPID_UNDEFINED, CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfo(), CFE_ES_ResetCFE(), CFE_ES_RestartApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_MATEEntry_t::Cooldown, HS_AMTEntry_t::CycleCount, HS_MATEEntry_t::EnableState, CFE_E_S_AppInfo::ExecutionCounter, HS_AMT_ACT_APP_RESTART, HS_AMT_ACT_EVENT, HS_AMT_ACT_LAST_NO_NMSG, HS_AMT_ACT_NOACT, HS_AMT_ACT_PROC_RESET, HS_AppData, HS_APPMON_APPNAME_DBG_EID, HS_APPMON_APPNAME_ERR_EID, HS_APPMON_FAIL_ERR_EID, HS_APPMON_MSGACTS_ERR_EID, HS_APPMON_NOT_RESTARTED_ERR_EID, HS_APPMON_PROC_ERR_EID, HS_APPMON_RESTART_ERR_EID, HS_BITS_PER_APPMON_ENABLE, HS_MAT_STATE_DISABLED, HS_MAT_STATE_NOEVENT, HS_MAX_MONITORED_APPS, HS_MAX_MSG_ACT_TYPES, HS_RESET_LIMIT_ERR_EID, HS_RESET_TASK_DELAY, HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATablePtr, HS_CDSData_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActCooldown, HS_AppData_t::MsgActExec, HS_AppData_t::MsgActsState, HS_MATEEntry_t::MsgBuf, OS_TaskDelay(), HS_CDSData_t::ResetsPerformed, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_ProcessMain().

Here is the call graph for this function:



12.12.2.2 HS_MonitorEvent()

```
void HS_MonitorEvent (
    const CFE_EVS_LongEventTlm_t * EventPtr )
```

Search the EventMon table for matches to the incoming event.

Description

Searches the Event Monitor Table for matches to the incoming event message. If a match is found, the table specified action is taken.

Assumptions, External Events, and Notes:

None

Parameters

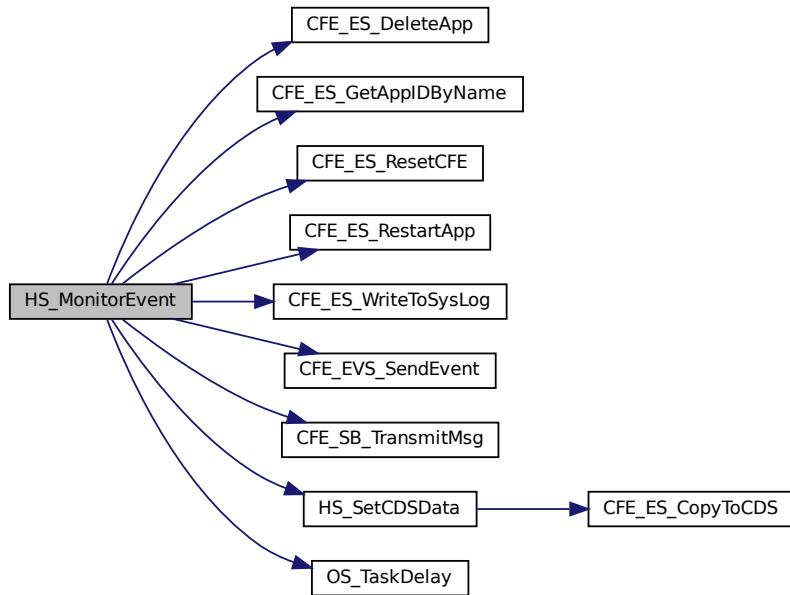
in	<code>EventPtr</code>	Pointer to the event message
----	-----------------------	------------------------------

Definition at line 229 of file hs_monitors.c.

References HS_EMTEntry_t::ActionType, HS_EMTEntry_t::AppName, CFE_EVS_PacketID::AppName, HS_AppData_t::CDSData, CFE_ES_APPID_UNDEFINED, CFE_ES_DeleteApp(), CFE_ES_GetAppIDByName(), CFE_ES_ResetCFE(), CFE_ES_RestartApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_MATEEntry_t::Cooldown, HS_AppData_t::EMTablePtr, HS_MATEEntry_t::EnableState, HS_EMTEntry_t::EventID, CFE_EVS_PacketID::EventID, HS_AMT_ACT_LAST_NONMSG, HS_AppData, HS_EMT_ACT_APP_DELETE, HS_EMT_ACT_APP_RESTART, HS_EMT_ACT_NOACT, HS_EMT_ACT_PROC_RESET, HS_EVENTMON_DELETE_ERR_EID, HS_EVENTMON_MSGACTS_ERR_EID, HS_EVENTMON_NOT_DELETED_ERR_EID, HS_EVENTMON_NOT_RESTARTED_ERR_EID, HS_EVENTMON_PROC_ERR_EID, HS_EVENTMON_RESTART_ERR_EID, HS_MAT_STATE_DISABLED, HS_MAT_STATE_NOEVENT, HS_MAX_MONITORED_EVENTS, HS_MAX_MSG_ACT_TYPES, HS_RESET_LIMIT_ERR_EID, HS_RESET_TASK_DELAY, HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATablePtr, HS_CDSData_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActCooldown, HS_AppData_t::MsgActExec, HS_AppData_t::MsgActsState, HS_MATEEntry_t::MsgBuf, OS_MAX_API_NAME, OS_TaskDelay(), CFE_EVS_LongEventTlm_Payload::PacketID, CFE_EVS_LongEventTlm::Payload, HS_CDSData_t::ResetsPerformed, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_ProcessCommands().

Here is the call graph for this function:



12.12.2.3 HS_MonitorUtilization()

```
void HS_MonitorUtilization (
    void )
```

Monitor the utilization tracker counter.

Description

Monitors the utilization tracker counter incremented by the Idle Task, converting it into an estimated CPU Utilization for the previous cycle. If the utilization is over a certain threshold for a certain amount of time, an event is output.

Assumptions, External Events, and Notes:

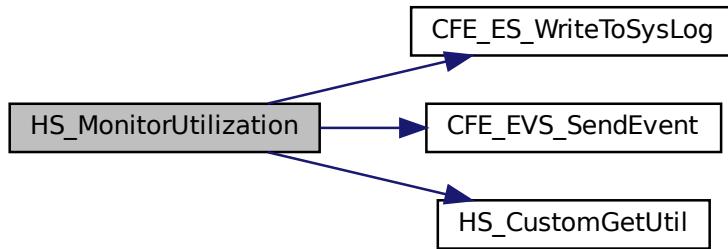
None

Definition at line 387 of file hs_monitors.c.

References CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), HS_AppData_t::CurrentCPUHoggingTime, HS_AppData_t::CurrentCPUHogState, HS_AppData_t::CurrentCPUUtilIndex, HS_AppData, HS_CPUMON_HOGGING_ERR_EID, HS_CustomGetUtil(), HS_STATE_ENABLED, HS_UTIL_AVERAGE_NUM_INTERVAL, HS_UTIL_PEAK_NUM_INTERVAL, HS_UTIL_PER_INTERVAL_HOGGING, HS_UTIL_PER_INTERVAL_TOTAL, HS_AppData_t::MaxCPUHoggingTime, HS_AppData_t::UtilCpuAvg, HS_AppData_t::UtilCpuPeak, and HS_AppData_t::UtilizationTracker.

Referenced by HS_CustomMonitorUtilization().

Here is the call graph for this function:

**12.12.2.4 HS_SetCDSData()**

```
void HS_SetCDSData (
    uint16 ResetsPerformed,
    uint16 MaxResets )
```

Update and store CDS data.

Description

This function is called to update and then store the data in the critical data store.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>ResetsPerformed</i>	Number of HS caused processor resets
in	<i>MaxResets</i>	Max number of resets allowed

Definition at line 849 of file hs_monitors.c.

References HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_CopyToCDS(), HS_AppData, HS_STA←TE_ENABLED, HS_CDSData_t::MaxResets, HS_CDSData_t::MaxResetsNot, HS_AppData_t::MyCDSHandle, HS_CDSData_t::ResetsPerformed, and HS_CDSData_t::ResetsPerformedNot.

Referenced by HS_AppInit(), HS_MonitorApplications(), HS_MonitorEvent(), HS_ResetResetsPerformedCmd(), and HS_SetMaxResetsCmd().

Here is the call graph for this function:

**12.12.2.5 HS_ValidateAMTable()**

```
int32 HS_ValidateAMTable (
    void * TableData )
```

Validate application monitor table.

Description

This function is called by table services when a validation of the application monitor table is required

Assumptions, External Events, and Notes:

None

Parameters

in	* <i>TableData</i>	Pointer to the table data to validate
----	--------------------	---------------------------------------

Returns

Table validation status

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>HS_AMTVAL_ERR_ACT</i>	Invalid ActionType specified.
<i>HS_AMTVAL_ERR_NUL</i>	Null Safety Buffer not Null.

See also

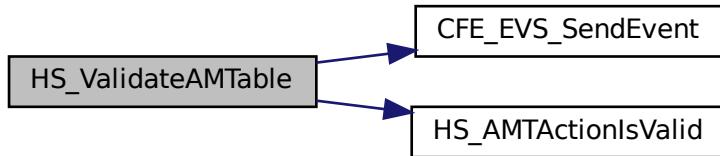
[HS_ValidateEMTable](#), [HS_ValidateXCTable](#), [HS_ValidateMATable](#)

Definition at line 467 of file hs_monitors.c.

References HS_AMTEntry_t::ActionType, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, HS_AMTEntry_t::CycleCount, HS_AM_TBL_NULL_ERR_EID, HS_AMT_ACT_NOACT, HS_AMTActionIsValid(), HS_AMTVAL_ERR_ACT, HS_AMTVAL_ERR_EID, HS_AMTVAL_ERR_NUL, HS_AMTVAL_INF_EID, HS_AMTVAL_NO_ERR, HS_MAX_MONITORED_APPS, HS_TBL_VAL_ERR, HS_AMTEEntry_t::NullTerm, and OS_MAX_API_NAME.

Referenced by HS_TblInit().

Here is the call graph for this function:

**12.12.2.6 HS_ValidateEMTable()**

```
int32 HS_ValidateEMTable (
    void * TableData )
```

Validate event monitor table.

Description

This function is called by table services when a validation of the event monitor table is required

Assumptions, External Events, and Notes:

None

Parameters

in	<i>*TableData</i>	Pointer to the table data to validate
----	-------------------	---------------------------------------

Returns

Table validation status

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>HS_EMTVAL_ERR_ACT</i>	Invalid ActionType specified.
<i>HS_EMTVAL_ERR_NUL</i>	Null Safety Buffer not Null.

See also

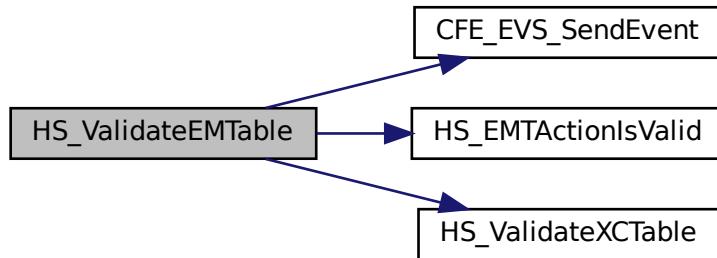
[HS_ValidateAMTable](#), [HS_ValidateXCTable](#), [HS_ValidateMATable](#)

Definition at line 560 of file hs_monitors.c.

References HS_EMTEEntry_t::ActionType, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, HS_EMTEEntry_t::EventID, HS_EM_TBL_NULL_ERR_EID, HS_EMT_ACT_NOACT, HS_EMTActionIsValid(), HS_EMTVAL_ERR_ACT, HS_EMTVAL_ERR_EID, HS_EMTVAL_ERR_NUL, HS_EMTVAL_INF_EID, HS_EMTVAL_NO_ERR, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_EVENTS, HS_TBL_VAL_ERR, HS_ValidateXCTable(), HS_XC_TBL_NULL_ERR_EID, HS_XCT_TYPE_APP_CHILD, HS_XCT_TYPE_APP_MAIN, HS_XCT_TYPE_DEVICE, HS_XCT_TYPE_ISR, HS_XCT_TYPE_NOTYPE, HS_XCTVAL_ERR_EID, HS_XCTVAL_ERR_NUL, HS_XCTVAL_ERR_TYPE, HS_XCTVAL_INF_EID, HS_XCTVAL_NO_ERR, HS_EMTEntry_t::NullTerm, HS_XCTEntry_t::NullTerm, OS_MAX_API_NAME, and HS_XCTEntry_t::ResourceType.

Referenced by HS_TblInit().

Here is the call graph for this function:



12.12.2.7 HS_ValidateMATable()

```
int32 HS_ValidateMATable (
    void * TableData )
```

Validate message actions table.

Description

This function is called by table services when a validation of the message actions table is required

Assumptions, External Events, and Notes:

None

Parameters

in	*TableData	Pointer to the table data to validate
----	------------	---------------------------------------

Returns

Table validation status

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
HS_MATVAL_ERR_ID	Invalid Message ID specified.
HS_MATVAL_ERR_LEN	Invalid Length specified.
HS_MATVAL_ERR_ENA	Invalid Enable State specified.

See also

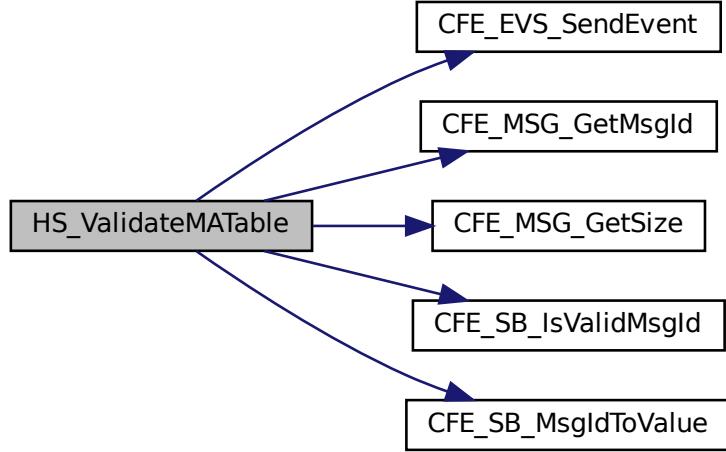
[HS_ValidateAMTable](#), [HS_ValidateEMTable](#), [HS_ValidateXCTable](#)

Definition at line 749 of file hs_monitors.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE↔_MISSION_SB_MAX_SB_MSG_SIZE, CFE_MSG_GetMsgId(), CFE_MSG_GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_IsValidMsgId(), CFE_SB_MsgIdToValue(), CFE_SUCCESS, HS_MATEEntry_t::EnableState, HS_MA_TBL↔_NULL_ERR_EID, HS_MAT_STATE_DISABLED, HS_MAT_STATE_ENABLED, HS_MAT_STATE_NOEVENT, HS↔_MATVAL_ERR_EID, HS_MATVAL_ERR_ENA, HS_MATVAL_ERR_ID, HS_MATVAL_ERR_LEN, HS_MATVAL_INF↔_EID, HS_MATVAL_NO_ERR, HS_MAX_MSG_ACT_TYPES, HS_TBL_VAL_ERR, and CFE_SB_Msg::Msg.

Referenced by HS_TblInit().

Here is the call graph for this function:



12.13 apps/hs/fsw/src/hs_monitors.h File Reference

```
#include "cfe.h"
#include "cfe_evs_msg.h"
```

Functions

- void [HS_MonitorApplications](#) (void)
Check execution status of each app in AppMon table.
- void [HS_MonitorEvent](#) (const [CFE_EVS_LongEventTlm_t](#) *EventPtr)
Search the EventMon table for matches to the incoming event.
- void [HS_MonitorUtilization](#) (void)
Monitor the utilization tracker counter.
- int32 [HS_ValidateAMTable](#) (void *TableData)
Validate application monitor table.
- int32 [HS_ValidateEMTable](#) (void *TableData)
Validate event monitor table.
- int32 [HS_ValidateXCTable](#) (void *TableData)
Validate execution counter table.
- int32 [HS_ValidateMATable](#) (void *TableData)
Validate message actions table.
- void [HS_SetCDSData](#) (uint16 ResetsPerformed, uint16 MaxResets)
Update and store CDS data.

12.13.1 Detailed Description

Specification for the CFS Health and Safety (HS) routines that handle application and event monitoring

12.13.2 Function Documentation

12.13.2.1 HS_MonitorApplications()

```
void HS_MonitorApplications (
    void )
```

Check execution status of each app in AppMon table.

Description

Cycles through the Application Monitor Table checking the current execution count for each monitored application. If the count fails to increment for the table specified duration, the table specified action is taken.

Assumptions, External Events, and Notes:

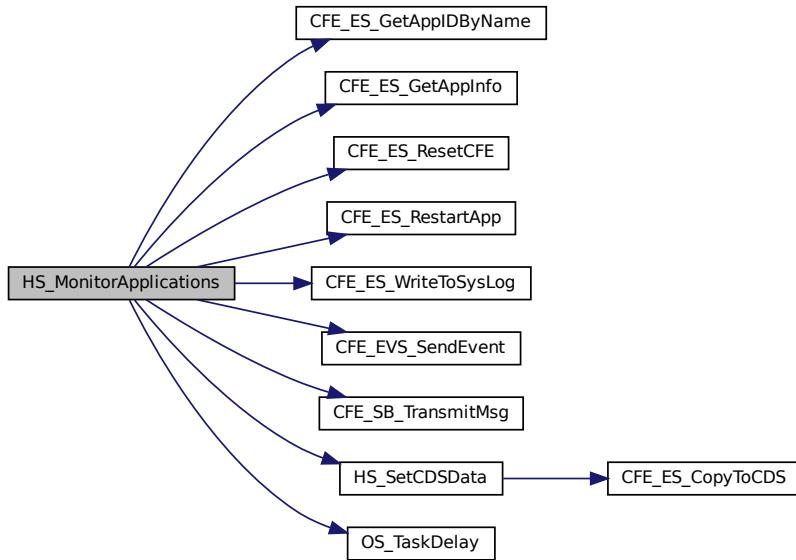
None

Definition at line 44 of file hs_monitors.c.

References HS_AMTEntry_t::ActionType, HS_AppData_t::AMTablePtr, HS_AppData_t::AppMonCheckInCountdown, HS_AppData_t::AppMonEnables, HS_AppData_t::AppMonLastExeCount, HS_AMTEntry_t::AppName, HS_AppData_t::CDSData, CFE_CLR, CFE_ES_APPID_UNDEFINED, CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfo(), CFE_ES_ResetCFE(), CFE_ES_RestartApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_MATEEntry_t::Cooldown, HS_AMTEntry_t::CycleCount, HS_MATEEntry_t::EnableState, CFE_E_S_AppInfo::ExecutionCounter, HS_AMT_ACT_APP_RESTART, HS_AMT_ACT_EVENT, HS_AMT_ACT_LAST_NO_NMSG, HS_AMT_ACT_NOACT, HS_AMT_ACT_PROC_RESET, HS_AppData, HS_APPMON_APPNAME_DBG_EID, HS_APPMON_APPNAME_ERR_EID, HS_APPMON_FAIL_ERR_EID, HS_APPMON_MSGACTS_ERR_EID, HS_APPMON_NOT_RESTARTED_ERR_EID, HS_APPMON_PROC_ERR_EID, HS_APPMON_RESTART_ERR_EID, HS_BITS_PER_APPMON_ENABLE, HS_MAT_STATE_DISABLED, HS_MAT_STATE_NOEVENT, HS_MAX_MONI_TORED_APPS, HS_MAX_MSG_ACT_TYPES, HS_RESET_LIMIT_ERR_EID, HS_RESET_TASK_DELAY, HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATablePtr, HS_CDSData_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActCooldown, HS_AppData_t::MsgActExec, HS_AppData_t::MsgActsState, HS_MATEEntry_t::MsgBuf, OS_TaskDelay(), HS_CDSData_t::ResetsPerformed, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_ProcessMain().

Here is the call graph for this function:



12.13.2.2 HS_MonitorEvent()

```
void HS_MonitorEvent (
    const CFE_EVS_LongEventTlm_t * EventPtr )
```

Search the EventMon table for matches to the incoming event.

Description

Searches the Event Monitor Table for matches to the incoming event message. If a match is found, the table specified action is taken.

Assumptions, External Events, and Notes:

None

Parameters

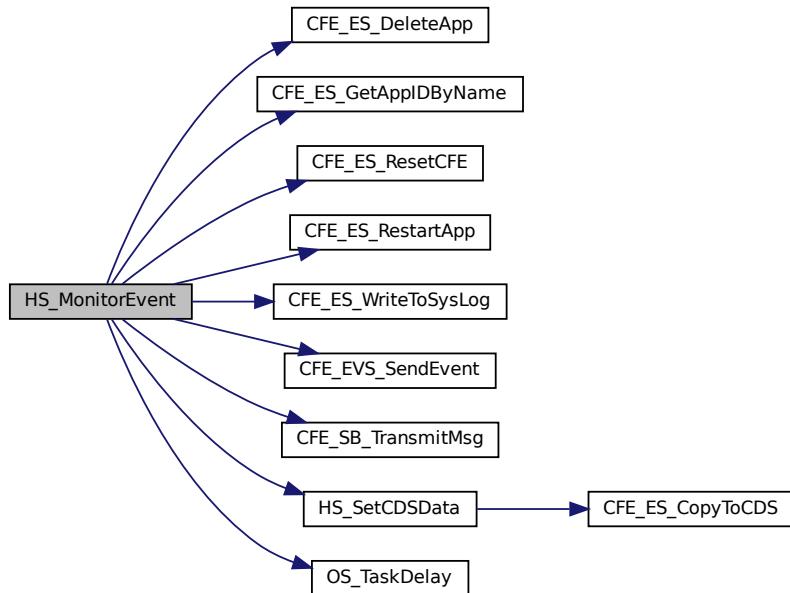
in	<code>EventPtr</code>	Pointer to the event message
----	-----------------------	------------------------------

Definition at line 229 of file hs_monitors.c.

References HS_EMTEntry_t::ActionType, HS_EMTEntry_t::AppName, CFE_EVS_PacketID::AppName, HS_AppData_t::CDSData, CFE_ES_APPID_UNDEFINED, CFE_ES_DeleteApp(), CFE_ES_GetAppIDByName(), CFE_ES_ResetCFE(), CFE_ES_RestartApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SB_TransmitMsg(), CFE_SUCCESS, HS_MATEEntry_t::Cooldown, HS_AppData_t::EMTablePtr, HS_MATEEntry_t::EnableState, HS_EMTEntry_t::EventID, CFE_EVS_PacketID::EventID, HS_AMT_ACT_LAST_NONMSG, HS_AppData, HS_EMT_ACT_APP_DELETE, HS_EMT_ACT_APP_RESTART, HS_EMT_ACT_NOACT, HS_EMT_ACT_PROC_RESET, HS_EVENTMON_DELETE_ERR_EID, HS_EVENTMON_MSGACTS_ERR_EID, HS_EVENTMON_NOT_DELETED_ERR_EID, HS_EVENTMON_NOT_RESTARTED_ERR_EID, HS_EVENTMON_PROC_ERR_EID, HS_EVENTMON_RESTART_ERR_EID, HS_MAT_STATE_DISABLED, HS_MAT_STATE_NOEVENT, HS_MAX_MONITORED_EVENTS, HS_MAX_MSG_ACT_TYPES, HS_RESET_LIMIT_ERR_EID, HS_RESET_TASK_DELAY, HS_SetCDSData(), HS_STATE_DISABLED, HS_STATE_ENABLED, HS_AppData_t::MATablePtr, HS_CDSData_t::MaxResets, CFE_SB_Msg::Msg, HS_AppData_t::MsgActCooldown, HS_AppData_t::MsgActExec, HS_AppData_t::MsgActsState, HS_MATEEntry_t::MsgBuf, OS_MAX_API_NAME, OS_TaskDelay(), CFE_EVS_LongEventTlm_Payload::PacketID, CFE_EVS_LongEventTlm::Payload, HS_CDSData_t::ResetsPerformed, and HS_AppData_t::ServiceWatchdogFlag.

Referenced by HS_ProcessCommands().

Here is the call graph for this function:



12.13.2.3 HS_MonitorUtilization()

```
void HS_MonitorUtilization (
    void )
```

Monitor the utilization tracker counter.

Description

Monitors the utilization tracker counter incremented by the Idle Task, converting it into an estimated CPU Utilization for the previous cycle. If the utilization is over a certain threshold for a certain amount of time, an event is output.

Assumptions, External Events, and Notes:

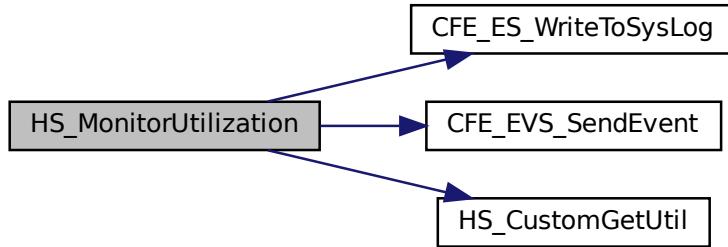
None

Definition at line 387 of file hs_monitors.c.

References CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), HS_AppData_t::CurrentCPUHoggingTime, HS_AppData_t::CurrentCPUHogState, HS_AppData_t::CurrentCPUUtilIndex, HS_AppData, HS_CPUMON_HOGGING_ERR_EID, HS_CustomGetUtil(), HS_STATE_ENABLED, HS_UTIL_AVERAGE_NUM_INTERVAL, HS_UTIL_PEAK_NUM_INTERVAL, HS_UTIL_PER_INTERVAL_HOGGING, HS_UTIL_PER_INTERVAL_TOTAL, HS_AppData_t::MaxCPUHoggingTime, HS_AppData_t::UtilCpuAvg, HS_AppData_t::UtilCpuPeak, and HS_AppData_t::UtilizationTracker.

Referenced by HS_CustomMonitorUtilization().

Here is the call graph for this function:

**12.13.2.4 HS_SetCDSData()**

```

void HS_SetCDSData (
    uint16 ResetsPerformed,
    uint16 MaxResets )
  
```

Update and store CDS data.

Description

This function is called to update and then store the data in the critical data store.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>ResetsPerformed</i>	Number of HS caused processor resets
in	<i>MaxResets</i>	Max number of resets allowed

Definition at line 849 of file hs_monitors.c.

References HS_AppData_t::CDSData, HS_AppData_t::CDSState, CFE_ES_CopyToCDS(), HS_AppData, HS_STA←TE_ENABLED, HS_CDSData_t::MaxResets, HS_CDSData_t::MaxResetsNot, HS_AppData_t::MyCDSHandle, HS_CDSData_t::ResetsPerformed, and HS_CDSData_t::ResetsPerformedNot.

Referenced by HS_AppInit(), HS_MonitorApplications(), HS_MonitorEvent(), HS_ResetResetsPerformedCmd(), and HS_SetMaxResetsCmd().

Here is the call graph for this function:

**12.13.2.5 HS_ValidateAMTable()**

```
int32 HS_ValidateAMTable (
    void * TableData )
```

Validate application monitor table.

Description

This function is called by table services when a validation of the application monitor table is required

Assumptions, External Events, and Notes:

None

Parameters

in	<i>*TableData</i>	Pointer to the table data to validate
----	-------------------	---------------------------------------

Returns

Table validation status

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>HS_AMTVAL_ERR_ACT</i>	Invalid ActionType specified.
<i>HS_AMTVAL_ERR_NUL</i>	Null Safety Buffer not Null.

See also

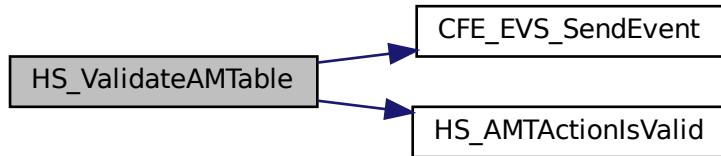
[HS_ValidateEMTable](#), [HS_ValidateXCTable](#), [HS_ValidateMATable](#)

Definition at line 467 of file hs_monitors.c.

References HS_AMTEntry_t::ActionType, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, HS_AMTEntry_t::CycleCount, HS_AM_TBL_NULL_ERR_EID, HS_AMT_ACT_NOACT, HS_AMTActionIsValid(), HS_AMTVAL_ERR_ACT, HS_AMTVAL_ERR_EID, HS_AMTVAL_ERR_NUL, HS_AMTVAL_INF_EID, HS_AMTVAL_NO_ERR, HS_MAX_MONITORED_APPS, HS_TBL_VAL_ERR, HS_AMTEEntry_t::NullTerm, and OS_MAX_API_NAME.

Referenced by HS_TblInit().

Here is the call graph for this function:

**12.13.2.6 HS_ValidateEMTable()**

```
int32 HS_ValidateEMTable (
    void * TableData )
```

Validate event monitor table.

Description

This function is called by table services when a validation of the event monitor table is required

Assumptions, External Events, and Notes:

None

Parameters

in	<i>*TableData</i>	Pointer to the table data to validate
----	-------------------	---------------------------------------

Returns

Table validation status

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>HS_EMTVAL_ERR_ACT</i>	Invalid ActionType specified.
<i>HS_EMTVAL_ERR_NUL</i>	Null Safety Buffer not Null.

See also

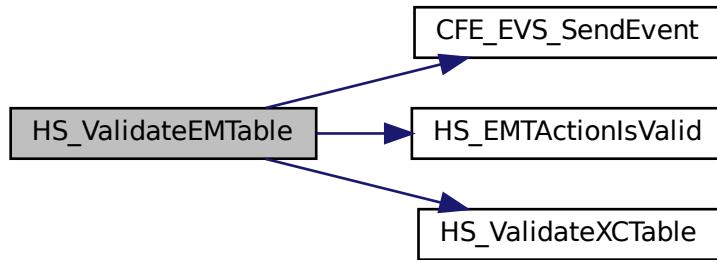
[HS_ValidateAMTable](#), [HS_ValidateXCTable](#), [HS_ValidateMATable](#)

Definition at line 560 of file hs_monitors.c.

References HS_EMTEEntry_t::ActionType, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, HS_EMTEEntry_t::EventID, HS_EM_TBL_NULL_ERR_EID, HS_EMT_ACT_NOACT, HS_EMTActionIsValid(), HS_EMTVAL_ERR_ACT, HS_EMTVAL_ERR_EID, HS_EMTVAL_ERR_NUL, HS_EMTVAL_INF_EID, HS_EMTVAL_NO_ERR, HS_MAX_EXEC_CNT_SLOTS, HS_MAX_MONITORED_EVENTS, HS_TBL_VAL_ERR, HS_ValidateXCTable(), HS_XC_TBL_NULL_ERR_EID, HS_XCT_TYPE_APP_CHILD, HS_XCT_TYPE_APP_MAIN, HS_XCT_TYPE_DEVICE, HS_XCT_TYPE_ISR, HS_XCT_TYPE_NOTYPE, HS_XCTVAL_ERR_EID, HS_XCTVAL_ERR_NUL, HS_XCTVAL_ERR_TYPE, HS_XCTVAL_INF_EID, HS_XCTVAL_NO_ERR, HS_EMTEntry_t::NullTerm, HS_XCTEntry_t::NullTerm, OS_MAX_API_NAME, and HS_XCTEntry_t::ResourceType.

Referenced by HS_TblInit().

Here is the call graph for this function:



12.13.2.7 HS_ValidateMATable()

```
int32 HS_ValidateMATable (
    void * TableData )
```

Validate message actions table.

Description

This function is called by table services when a validation of the message actions table is required

Assumptions, External Events, and Notes:

None

Parameters

in	*TableData	Pointer to the table data to validate
----	------------	---------------------------------------

Returns

Table validation status

Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
<i>HS_MATVAL_ERR_ID</i>	Invalid Message ID specified.
<i>HS_MATVAL_ERR_LEN</i>	Invalid Length specified.
<i>HS_MATVAL_ERR_ENA</i>	Invalid Enable State specified.

See also

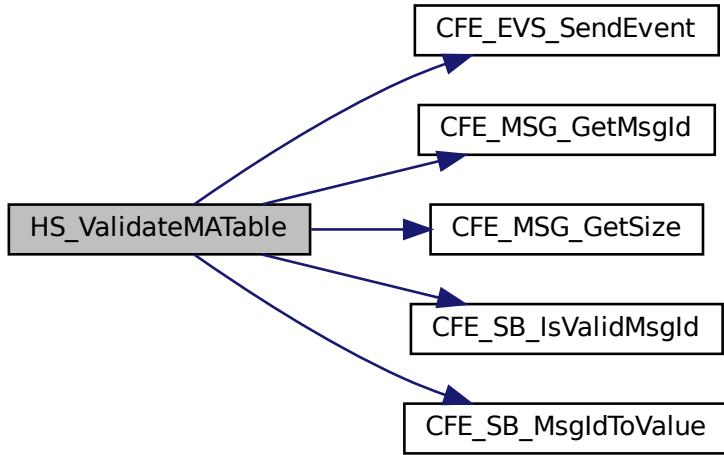
[HS_ValidateAMTable](#), [HS_ValidateEMTable](#), [HS_ValidateXCTable](#)

Definition at line 749 of file hs_monitors.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MMISSION_SB_MAX_SB_MSG_SIZE, CFE_MSG_GetMsgId(), CFE_MSG_GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_IsValidMsgId(), CFE_SB_MsgIdToValue(), CFE_SUCCESS, HS_MATEEntry_t::EnableState, HS_MA_TBL_NULL_ERR_EID, HS_MAT_STATE_DISABLED, HS_MAT_STATE_ENABLED, HS_MAT_STATE_NOEVENT, HS_MATVAL_ERR_EID, HS_MATVAL_ERR_ENA, HS_MATVAL_ERR_ID, HS_MATVAL_ERR_LEN, HS_MATVAL_INF_EID, HS_MATVAL_NO_ERR, HS_MAX_MSG_ACT_TYPES, HS_TBL_VAL_ERR, and CFE_SB_Msg::Msg.

Referenced by HS_TblInit().

Here is the call graph for this function:



12.13.2.8 HS_ValidateXCTable()

```
int32 HS_ValidateXCTable (
    void * TableData )
```

Validate execution counter table.

Description

This function is called by table services when a validation of the execution counter table is required

Assumptions, External Events, and Notes:

None

Parameters

in	* <i>TableData</i>	Pointer to the table data to validate
----	--------------------	---------------------------------------

Returns

Table validation status

Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>HS_XCTVAL_ERR_TYPE</code>	Invalid Counter Type specified.
<code>HS_XCTVAL_ERR_NUL</code>	Null Safety Buffer not Null.

See also

[HS_ValidateAMTable](#), [HS_ValidateEMTable](#), [HS_ValidateMATable](#)

Referenced by `HS_TblInit()`, and `HS_ValidateEMTable()`.

12.14 apps/hs/fsw/src/hs_msg.h File Reference

```
#include "hs_msgdefs.h"
#include "hs_platform_cfg.h"
#include "cfe.h"
```

Data Structures

- struct [HS_NoArgsCmd_t](#)
No Arguments Command.
- struct [HS_SetMaxResetsCmd_t](#)
Set Max Resets Command.
- struct [HS_HkPacket_t](#)
Housekeeping Packet Structure.

Macros

- #define `HS_BITS_PER_APPMON_ENABLE` 32
HS Bits per AppMon Enable entry.

12.14.1 Detailed Description

Specification for the CFS Health and Safety (HS) command and telemetry message data types.

Note

Constants and enumerated types related to these message structures are defined in [hs_msgdefs.h](#). They are kept separate to allow easy integration with ASIST RDL files which can't handle typedef declarations (see the main comment block in [hs_msgdefs.h](#) for more info).

12.14.2 Macro Definition Documentation

12.14.2.1 HS_BITS_PER_APPMON_ENABLE

```
#define HS_BITS_PER_APPMON_ENABLE 32
```

HS Bits per AppMon Enable entry.

Definition at line 46 of file hs_msg.h.

Referenced by HS_AppMonStatusRefresh(), HS_HousekeepingReq(), and HS_MonitorApplications().

12.15 apps/fsf/src/hs_msgdefs.h File Reference

Macros

- `#define HS_NOOP_CC 0`
Noop.
- `#define HS_RESET_CC 1`
Reset Counters.
- `#define HS_ENABLE_APPMON_CC 2`
Enable Applications Monitor.
- `#define HS_DISABLE_APPMON_CC 3`
Disable Applications Monitor.
- `#define HS_ENABLE_EVENTMON_CC 4`
Enable Events Monitor.
- `#define HS_DISABLE_EVENTMON_CC 5`
Disable Events Monitor.
- `#define HS_ENABLE_ALIVENESS_CC 6`
Enable Aliveness Indicator.
- `#define HS_DISABLE_ALIVENESS_CC 7`
Disable Aliveness Indicator.
- `#define HS_RESET_RESETS_PERFORMED_CC 8`
Reset Processor Resets Performed Count.
- `#define HS_SET_MAX_RESETS_CC 9`
Set Max Processor Resets Performed Count.
- `#define HS_ENABLE_CPUHOG_CC 10`
Enable CPU Hogging Indicator.
- `#define HS_DISABLE_CPUHOG_CC 11`
Disable CPU Hogging Indicator.

HS Switch States (AppMon, EventMon, Aliveness)

- `#define HS_STATE_DISABLED 0`
- `#define HS_STATE_ENABLED 1`

HS Internal Status Flags

- #define HS_LOADED_XCT 0x01
- #define HS_LOADED_MAT 0x02
- #define HS_LOADED_AMT 0x04
- #define HS_LOADED_EMT 0x08
- #define HS_CDS_IN_USE 0x10

HS Invalid Execution Counter

- #define HS_INVALID_EXECOUNT 0xFFFFFFFF

12.15.1 Detailed Description

Specification for the CFS Health and Safety (HS) command and telemetry message constant definitions.

Note

These Macro definitions have been put in this file (instead of `hs_msg.h`) so this file can be included directly into ASIST build test scripts. ASIST RDL files can accept C language defines but can't handle type definitions. As a result: DO NOT PUT ANY TYPEDEFS OR STRUCTURE DEFINITIONS IN THIS FILE! ADD THEM TO `hs_msg.h` IF NEEDED!

12.15.2 Macro Definition Documentation

12.15.2.1 HS_CDS_IN_USE

```
#define HS_CDS_IN_USE 0x10
```

Definition at line 56 of file `hs_msgdefs.h`.

Referenced by `HS_HousekeepingReq()`.

12.15.2.2 HS_INVALID_EXECOUNT

```
#define HS_INVALID_EXECOUNT 0xFFFFFFFF
```

Definition at line 63 of file `hs_msgdefs.h`.

Referenced by `HS_HousekeepingReq()`, and `HS_TblInit()`.

12.15.2.3 HS_LOADED_AMT

```
#define HS_LOADED_AMT 0x04
```

Definition at line 54 of file hs_msgdefs.h.

Referenced by HS_HousekeepingReq().

12.15.2.4 HS_LOADED_EMT

```
#define HS_LOADED_EMT 0x08
```

Definition at line 55 of file hs_msgdefs.h.

Referenced by HS_HousekeepingReq().

12.15.2.5 HS_LOADED_MAT

```
#define HS_LOADED_MAT 0x02
```

Definition at line 53 of file hs_msgdefs.h.

Referenced by HS_HousekeepingReq().

12.15.2.6 HS_LOADED_XCT

```
#define HS_LOADED_XCT 0x01
```

Definition at line 52 of file hs_msgdefs.h.

Referenced by HS_HousekeepingReq().

12.15.2.7 HS_STATE_DISABLED

```
#define HS_STATE_DISABLED 0
```

Definition at line 44 of file hs_msgdefs.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_EnableEventMonCmd(), HS_MonitorApplications(), HS_MonitorEvent(), and HS_TblInit().

12.15.2.8 HS_STATE_ENABLED

```
#define HS_STATE_ENABLED 1
```

Definition at line 45 of file hs_msgdefs.h.

Referenced by HS_AcquirePointers(), HS_AppInit(), HS_AppMain(), HS_DisableEventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_MonitorApplications(), HS_MonitorEvent(), HS_MonitorUtilization(), HS_ProcessCommands(), HS_ProcessMain(), and HS_SetCDSData().

12.16 apps/hs/fsw/src/hs_tbl.h File Reference

```
#include "cfe.h"
#include "hs_tbldefs.h"
#include "hs_platform_cfg.h"
```

Data Structures

- struct [HS_AMTEntry_t](#)
Application Monitor Table (AMT) Entry.
- struct [HS_EMTEEntry_t](#)
Event Monitor Table (EMT) Entry.
- struct [HS_XCTEntry_t](#)
Execution Counters Table (XCT) Entry.
- union [HS_MATMsgBuf_t](#)
Message Action Table buffer.
- struct [HS_MATEEntry_t](#)
Message Actions Table (MAT) Entry.

Macros

Macros for Action Type numbers of Message Actions

- #define [HS_AMT_ACT_MSG](#)(num) (HS_AMT_ACT_LAST_NONMSG + 1 + (num))
- #define [HS_EMT_ACT_MSG](#)(num) (HS_EMT_ACT_LAST_NONMSG + 1 + (num))

12.16.1 Detailed Description

Specification for the CFS Health and Safety (HS) table structures

Note

Constants and enumerated types related to these table structures are defined in [hs_tbldefs.h](#). They are kept separate to allow easy integration with ASIST RDL files which can't handle typedef declarations (see the main comment block in [hs_tbldefs.h](#) for more info).

12.16.2 Macro Definition Documentation

12.16.2.1 HS_AMT_ACT_MSG

```
#define HS_AMT_ACT_MSG( num ) (HS_AMT_ACT_LAST_NONMSG + 1 + (num))
```

Definition at line 49 of file hs_tbl.h.

12.16.2.2 HS_EMT_ACT_MSG

```
#define HS_EMT_ACT_MSG( num ) (HS_EMT_ACT_LAST_NONMSG + 1 + (num))
```

Definition at line 50 of file hs_tbl.h.

12.17 apps/hs/fsw/src/hs_tbldefs.h File Reference

Macros

HS Table Name Strings

- #define HS_AMT_TABLENAME "AppMon_Tbl"
AppMon Table Name.
- #define HS_EMT_TABLENAME "EventMon_Tbl"
EventMon Table Name.
- #define HS_XCT_TABLENAME "ExeCount_Tbl"
ExeCount Table Name.
- #define HS_MAT_TABLENAME "MsgActs_Tbl"
MsgAction Table Name.

Application Monitor Table (AMT) Action Types

- #define HS_AMT_ACT_NOACT 0
No action is taken.
- #define HS_AMT_ACT_PROC_RESET 1
Generates Processor Reset on failure.
- #define HS_AMT_ACT_APP_RESTART 2
Attempts to restart application on failure.
- #define HS_AMT_ACT_EVENT 3
Generates event message on failure.
- #define HS_AMT_ACT_LAST_NONMSG 3
Index for finding end of non-message actions.

Event Monitor Table (EMT) Action Types

- #define HS_EMT_ACT_NOACT 0
No action is taken.
- #define HS_EMT_ACT_PROC_RESET 1
Generates Processor Reset on detection.
- #define HS_EMT_ACT_APP_RESTART 2
Attempts to restart application on detection.
- #define HS_EMT_ACT_APP_DELETE 3
Deletes application on detection.
- #define HS_EMT_ACT_LAST_NONMSG 3
Index for finding end of non-message actions.

Execution Counters Table (XCT) Resource Types

- #define HS_XCT_TYPE_NOTYPE 0
No type.
- #define HS_XCT_TYPE_APP_MAIN 1
Counter for Application Main task.
- #define HS_XCT_TYPE_APP_CHILD 2
Counter for Application Child task.
- #define HS_XCT_TYPE_DEVICE 3
Counter for Device Driver.
- #define HS_XCT_TYPE_ISR 4
Counter for Interrupt Service Routine.

Message Actions Table (MAT) Enable State

- #define HS_MAT_STATE_DISABLED 0
Message Actions are Disabled.
- #define HS_MAT_STATE_ENABLED 1
Message Actions are Enabled.
- #define HS_MAT_STATE_NOEVENT 2
Message Actions are Enabled but produce no events.

Application Monitor Table (AMT) Validation Error Enumerated Types

- #define HS_AMTVAL_NO_ERR 0
No error.
- #define HS_AMTVAL_ERR_ACT -1
Invalid ActionType specified.
- #define HS_AMTVAL_ERR_NUL -2
Null Safety Buffer not Null.

Event Monitor Table (EMT) Validation Error Enumerated Types

- #define HS_EMTVAL_NO_ERR 0
No error.
- #define HS_EMTVAL_ERR_ACT -1
Invalid ActionType specified.
- #define HS_EMTVAL_ERR_NUL -2
Null Safety Buffer not Null.

Event Counter Table (XCT) Validation Error Enumerated Types

- #define HS_XCTVAL_NO_ERR 0
No error.
- #define HS_XCTVAL_ERR_TYPE -1
Invalid Counter Type specified.
- #define HS_XCTVAL_ERR_NUL -2
Null Safety Buffer not Null.

Message Actions Table (MAT) Validation Error Enumerated Types

- #define HS_MATVAL_NO_ERR 0
No error.
- #define HS_MATVAL_ERR_ID -1
Invalid Message ID specified.
- #define HS_MATVAL_ERR_LEN -2
Invalid Length specified.
- #define HS_MATVAL_ERR_ENA -3
Invalid Enable State specified.

12.17.1 Detailed Description

Specification for the CFS Health and Safety (HS) table related constant definitions.

Note

These Macro definitions have been put in this file (instead of [hs_tbl.h](#)) so this file can be included directly into ASIST build test scripts. ASIST RDL files can accept C language defines but can't handle type definitions. As a result: DO NOT PUT ANY TYPEDEFS OR STRUCTURE DEFINITIONS IN THIS FILE! ADD THEM TO [hs_tbl.h](#) IF NEEDED!

12.17.2 Macro Definition Documentation

12.17.2.1 HS_AMT_ACT_APP_RESTART

```
#define HS_AMT_ACT_APP_RESTART 2
```

Attempts to restart application on failure.

Definition at line 56 of file [hs_tbldefs.h](#).

Referenced by [HS_MonitorApplications\(\)](#).

12.17.2.2 HS_AMT_ACT_EVENT

```
#define HS_AMT_ACT_EVENT 3
```

Generates event message on failure.

Definition at line 57 of file hs_tbldefs.h.

Referenced by HS_MonitorApplications().

12.17.2.3 HS_AMT_ACT_LAST_NONMSG

```
#define HS_AMT_ACT_LAST_NONMSG 3
```

Index for finding end of non-message actions.

Definition at line 58 of file hs_tbldefs.h.

Referenced by HS_AMTActionIsValid(), HS_MonitorApplications(), and HS_MonitorEvent().

12.17.2.4 HS_AMT_ACT_NOACT

```
#define HS_AMT_ACT_NOACT 0
```

No action is taken.

Definition at line 54 of file hs_tbldefs.h.

Referenced by HS_AMTActionIsValid(), HS_AppMonStatusRefresh(), HS_MonitorApplications(), and HS_ValidateAMTable().

12.17.2.5 HS_AMT_ACT_PROC_RESET

```
#define HS_AMT_ACT_PROC_RESET 1
```

Generates Processor Reset on failure.

Definition at line 55 of file hs_tbldefs.h.

Referenced by HS_MonitorApplications().

12.17.2.6 HS_AMT_TABLENAME

```
#define HS_AMT_TABLENAME "AppMon_Tbl"
```

AppMon Table Name.

Definition at line 44 of file hs_tbldefs.h.

Referenced by HS_TblInit().

12.17.2.7 HS_AMTVAL_ERR_ACT

```
#define HS_AMTVAL_ERR_ACT -1
```

Invalid ActionType specified.

Definition at line 97 of file hs_tbldefs.h.

Referenced by HS_ValidateAMTable().

12.17.2.8 HS_AMTVAL_ERR_NUL

```
#define HS_AMTVAL_ERR_NUL -2
```

Null Safety Buffer not Null.

Definition at line 98 of file hs_tbldefs.h.

Referenced by HS_ValidateAMTable().

12.17.2.9 HS_AMTVAL_NO_ERR

```
#define HS_AMTVAL_NO_ERR 0
```

No error.

Definition at line 96 of file hs_tbldefs.h.

Referenced by HS_ValidateAMTable().

12.17.2.10 HS_EMT_ACT_APP_DELETE

```
#define HS_EMT_ACT_APP_DELETE 3
```

Deletes application on detection.

Definition at line 68 of file hs_tbldefs.h.

Referenced by HS_MonitorEvent().

12.17.2.11 HS_EMT_ACT_APP_RESTART

```
#define HS_EMT_ACT_APP_RESTART 2
```

Attempts to restart application on detection.

Definition at line 67 of file hs_tbldefs.h.

Referenced by HS_MonitorEvent().

12.17.2.12 HS_EMT_ACT_LAST_NONMSG

```
#define HS_EMT_ACT_LAST_NONMSG 3
```

Index for finding end of non-message actions.

Definition at line 69 of file hs_tbldefs.h.

Referenced by HS_EMTActionIsValid().

12.17.2.13 HS_EMT_ACT_NOACT

```
#define HS_EMT_ACT_NOACT 0
```

No action is taken.

Definition at line 65 of file hs_tbldefs.h.

Referenced by HS_EMTActionIsValid(), HS_HousekeepingReq(), HS_MonitorEvent(), and HS_ValidateEMTable().

12.17.2.14 HS_EMT_ACT_PROC_RESET

```
#define HS_EMT_ACT_PROC_RESET 1
```

Generates Processor Reset on detection.

Definition at line 66 of file hs_tbldefs.h.

Referenced by HS_MonitorEvent().

12.17.2.15 HS_EMT_TABLENAME

```
#define HS_EMT_TABLENAME "EventMon_Tbl"
```

EventMon Table Name.

Definition at line 45 of file hs_tbldefs.h.

Referenced by HS_TblInit().

12.17.2.16 HS_EMTVAL_ERR_ACT

```
#define HS_EMTVAL_ERR_ACT -1
```

Invalid ActionType specified.

Definition at line 106 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.17.2.17 HS_EMTVAL_ERR_NUL

```
#define HS_EMTVAL_ERR_NUL -2
```

Null Safety Buffer not Null.

Definition at line 107 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.17.2.18 HS_EMTVAL_NO_ERR

```
#define HS_EMTVAL_NO_ERR 0
```

No error.

Definition at line 105 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.17.2.19 HS_MAT_STATE_DISABLED

```
#define HS_MAT_STATE_DISABLED 0
```

Message Actions are Disabled.

Definition at line 87 of file hs_tbldefs.h.

Referenced by HS_MonitorApplications(), HS_MonitorEvent(), and HS_ValidateMATable().

12.17.2.20 HS_MAT_STATE_ENABLED

```
#define HS_MAT_STATE_ENABLED 1
```

Message Actions are Enabled.

Definition at line 88 of file hs_tbldefs.h.

Referenced by HS_ValidateMATable().

12.17.2.21 HS_MAT_STATE_NOEVENT

```
#define HS_MAT_STATE_NOEVENT 2
```

Message Actions are Enabled but produce no events.

Definition at line 89 of file hs_tbldefs.h.

Referenced by HS_MonitorApplications(), HS_MonitorEvent(), and HS_ValidateMATable().

12.17.2.22 HS_MAT_TABLENAME

```
#define HS_MAT_TABLENAME "MsgActs_Tbl"
```

MsgAction Table Name.

Definition at line 47 of file hs_tbldefs.h.

Referenced by HS_TblInit().

12.17.2.23 HS_MATVAL_ERR_ENA

```
#define HS_MATVAL_ERR_ENA -3
```

Invalid Enable State specified.

Definition at line 126 of file hs_tbldefs.h.

Referenced by HS_ValidateMATable().

12.17.2.24 HS_MATVAL_ERR_ID

```
#define HS_MATVAL_ERR_ID -1
```

Invalid Message ID specified.

Definition at line 124 of file hs_tbldefs.h.

Referenced by HS_ValidateMATable().

12.17.2.25 HS_MATVAL_ERR_LEN

```
#define HS_MATVAL_ERR_LEN -2
```

Invalid Length specified.

Definition at line 125 of file hs_tbldefs.h.

Referenced by HS_ValidateMATable().

12.17.2.26 HS_MATVAL_NO_ERR

```
#define HS_MATVAL_NO_ERR 0
```

No error.

Definition at line 123 of file hs_tbldefs.h.

Referenced by HS_ValidateMATable().

12.17.2.27 HS_XCT_TABLENAME

```
#define HS_XCT_TABLENAME "ExeCount_Tbl"
```

ExeCount Table Name.

Definition at line 46 of file hs_tbldefs.h.

Referenced by HS_TblInit().

12.17.2.28 HS_XCT_TYPE_APP_CHILD

```
#define HS_XCT_TYPE_APP_CHILD 2
```

Counter for Application Child task.

Definition at line 78 of file hs_tbldefs.h.

Referenced by HS_HousekeepingReq(), and HS_ValidateEMTable().

12.17.2.29 HS_XCT_TYPE_APP_MAIN

```
#define HS_XCT_TYPE_APP_MAIN 1
```

Counter for Application Main task.

Definition at line 77 of file hs_tbldefs.h.

Referenced by HS_HousekeepingReq(), and HS_ValidateEMTable().

12.17.2.30 HS_XCT_TYPE_DEVICE

```
#define HS_XCT_TYPE_DEVICE 3
```

Counter for Device Driver.

Definition at line 79 of file hs_tbldefs.h.

Referenced by HS_HousekeepingReq(), and HS_ValidateEMTable().

12.17.2.31 HS_XCT_TYPE_ISR

```
#define HS_XCT_TYPE_ISR 4
```

Counter for Interrupt Service Routine.

Definition at line 80 of file hs_tbldefs.h.

Referenced by HS_HousekeepingReq(), and HS_ValidateEMTable().

12.17.2.32 HS_XCT_TYPE_NOTYPE

```
#define HS_XCT_TYPE_NOTYPE 0
```

No type.

Definition at line 76 of file hs_tbldefs.h.

Referenced by HS_HousekeepingReq(), and HS_ValidateEMTable().

12.17.2.33 HS_XCTVAL_ERR_NUL

```
#define HS_XCTVAL_ERR_NUL -2
```

Null Safety Buffer not Null.

Definition at line 116 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.17.2.34 HS_XCTVAL_ERR_TYPE

```
#define HS_XCTVAL_ERR_TYPE -1
```

Invalid Counter Type specified.

Definition at line 115 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.17.2.35 HS_XCTVAL_NO_ERR

```
#define HS_XCTVAL_NO_ERR 0
```

No error.

Definition at line 114 of file hs_tbldefs.h.

Referenced by HS_ValidateEMTable().

12.18 apps/hs/fsw/src/hs_utils.c File Reference

```
#include "hs_app.h"
#include "hs_utils.h"
#include "hs_custom.h"
#include "hs_monitors.h"
#include "hs_msgids.h"
#include "hs_events.h"
#include "hs_version.h"
```

Functions

- bool [HS_VerifyMsgLength](#) (const [CFE_MSG_Message_t](#) *MsgPtr, size_t ExpectedLength)
Verify message length.
- bool [HS_AMTActionsValid](#) (uint16 ActionType)
Verify AMT Action Type.
- bool [HS_EMTActionsValid](#) (uint16 ActionType)
Verify EMT Action Type.

12.18.1 Detailed Description

Utility functions for the CFS Health and Safety (HS) application.

12.18.2 Function Documentation

12.18.2.1 HS_AMTActionIsValid()

```
bool HS_AMTActionIsValid (
    uint16 ActionType )
```

Verify AMT Action Type.

Description

Checks if the specified value is a valid AMT Action Type.

Assumptions, External Events, and Notes:

None

Parameters

in	ActionType	Action type to validate
----	------------	-------------------------

Returns

Boolean action valid response

Return values

true	Action type valid
false	Action type not valid

Definition at line 94 of file hs_utils.c.

References HS_AMT_ACT_LAST_NONMSG, HS_AMT_ACT_NOACT, and HS_MAX_MSG_ACT_TYPES.

Referenced by HS_ValidateAMTable().

12.18.2.2 HS_EMTActionIsValid()

```
bool HS_EMTActionIsValid (
    uint16 ActionType )
```

Verify EMT Action Type.

Description

Checks if the specified value is a valid EMT Action Type.

Assumptions, External Events, and Notes:

None

Parameters

in	ActionType	Action type to validate
----	------------	-------------------------

Returns

Boolean action valid response

Return values

true	Action type valid
false	Action type not valid

Definition at line 118 of file hs_utils.c.

References HS_EMT_ACT_LAST_NONMSG, HS_EMT_ACT_NOACT, and HS_MAX_MSG_ACT_TYPES.

Referenced by HS_ValidateEMTTable().

12.18.2.3 HS_VerifyMsgLength()

```
bool HS_VerifyMsgLength (
    const CFE_MSG_Message_t * MsgPtr,
    size_t ExpectedLength )
```

Verify message length.

Description

Checks if the actual length of a software bus message matches the expected length and sends an error event if a mismatch occurs

Assumptions, External Events, and Notes:

None

Parameters

in	<i>MsgPtr</i>	Message Pointer
in	<i>ExpectedLength</i>	Expected length

Returns

Boolean message length matches response

Return values

<i>true</i>	Length matches expected
<i>false</i>	Length does not match expected

See also

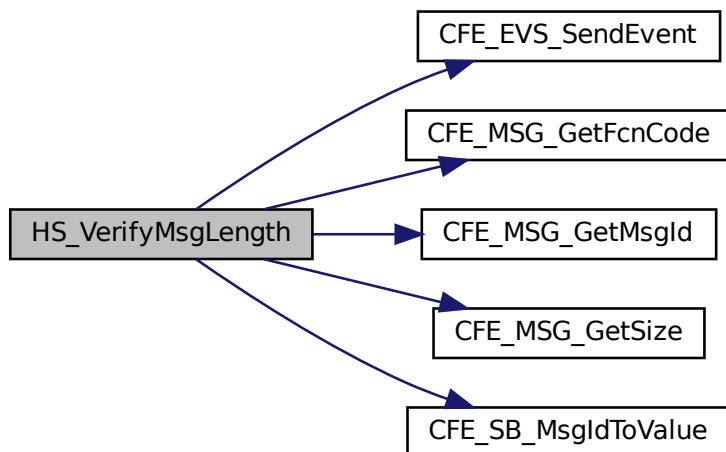
[HS_LEN_ERR_EID](#)

Definition at line 41 of file hs_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG_GetMsgId(), CFE_MSG_GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), HS_AppData_t::CmdErrCount, HS_AppData, HS_HKREQ_LEN_ERR_EID, HS_LEN_ERR_EID, and HS_SEND_HK_MID.

Referenced by HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_NoopCmd(), HS_ResetCmd(), HS_ResetResetsPerformedCmd(), HS_SetMaxResetsCmd(), HS_SetUtilDiagCmd(), and HS_SetUtilParamsCmd().

Here is the call graph for this function:



12.19 apps/hs/fsw/src/hs_utils.h File Reference

```
#include "cfe.h"
```

Functions

- bool [HS_VerifyMsgLength](#) (const [CFE_MSG_Message_t](#) *MsgPtr, size_t ExpectedLength)
Verify message length.
- bool [HS_AMTActionIsValid](#) (uint16 ActionType)
Verify AMT Action Type.
- bool [HS_EMTActionIsValid](#) (uint16 ActionType)
Verify EMT Action Type.

12.19.1 Detailed Description

Utility functions for the cFS Health and Safety (HS) application.

12.19.2 Function Documentation

12.19.2.1 [HS_AMTActionIsValid\(\)](#)

```
bool HS_AMTActionIsValid (
    uint16 ActionType )
```

Verify AMT Action Type.

Description

Checks if the specified value is a valid AMT Action Type.

Assumptions, External Events, and Notes:

None

Parameters

in	ActionType	Action type to validate
----	------------	-------------------------

Returns

Boolean action valid response

Return values

<i>true</i>	Action type valid
<i>false</i>	Action type not valid

Definition at line 94 of file hs_utils.c.

References HS_AMT_ACT_LAST_NONMSG, HS_AMT_ACT_NOACT, and HS_MAX_MSG_ACT_TYPES.

Referenced by HS_ValidateAMTable().

12.19.2.2 HS_EMTActionIsValid()

```
bool HS_EMTActionIsValid (
    uint16 ActionType )
```

Verify EMT Action Type.

Description

Checks if the specified value is a valid EMT Action Type.

Assumptions, External Events, and Notes:

None

Parameters

<i>in</i>	<i>ActionType</i>	Action type to validate
-----------	-------------------	-------------------------

Returns

Boolean action valid response

Return values

<i>true</i>	Action type valid
<i>false</i>	Action type not valid

Definition at line 118 of file hs_utils.c.

References HS_EMT_ACT_LAST_NONMSG, HS_EMT_ACT_NOACT, and HS_MAX_MSG_ACT_TYPES.

Referenced by HS_ValidateEMTable().

12.19.2.3 HS_VerifyMsgLength()

```
bool HS_VerifyMsgLength (
    const CFE_MSG_Message_t * MsgPtr,
    size_t ExpectedLength )
```

Verify message length.

Description

Checks if the actual length of a software bus message matches the expected length and sends an error event if a mismatch occurs

Assumptions, External Events, and Notes:

None

Parameters

in	<i>MsgPtr</i>	Message Pointer
in	<i>ExpectedLength</i>	Expected length

Returns

Boolean message length matches response

Return values

<i>true</i>	Length matches expected
<i>false</i>	Length does not match expected

See also

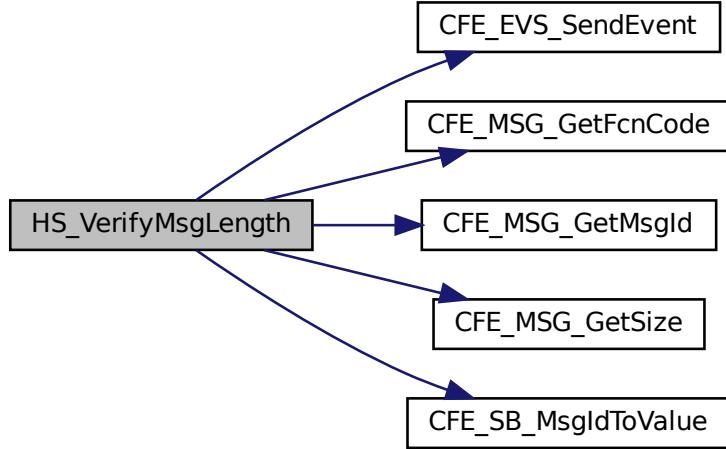
[HS_LEN_ERR_EID](#)

Definition at line 41 of file hs_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), CFE_MSG_GetMsgId(), CFE_MSG.GetSize(), CFE_SB_INVALID_MSG_ID, CFE_SB_MsgIdToValue(), HS_AppData_t::CmdErrCount, HS_AppData, HS_HKREQ_LEN_ERR_EID, HS_LEN_ERR_EID, and HS_SEND_HK_MID.

Referenced by HS_DisableAlivenessCmd(), HS_DisableAppMonCmd(), HS_DisableCPUHogCmd(), HS_DisableEventMonCmd(), HS_EnableAlivenessCmd(), HS_EnableAppMonCmd(), HS_EnableCPUHogCmd(), HS_EnableEventMonCmd(), HS_HousekeepingReq(), HS_NoopCmd(), HS_ResetCmd(), HS_ResetResetsPerformedCmd(), HS_SetMaxResetsCmd(), HS_SetUtilDiagCmd(), and HS_SetUtilParamsCmd().

Here is the call graph for this function:



12.20 apps/hs/fsw/src/hs_verify.h File Reference

12.20.1 Detailed Description

Contains CFS Health and Safety (HS) macros that run preprocessor checks on mission and platform configurable parameters

12.21 apps/hs/fsw/src/hs_version.h File Reference

Macros

- `#define HS_MAJOR_VERSION 2`
Major version number.
- `#define HS_MINOR_VERSION 4`
Minor version number.
- `#define HS_REVISION 99`
Revision number.

12.21.1 Detailed Description

Contains version tags for the Core Flight System (CFS) Health and Safety (HS) Application.

12.22 apps/hs/fsw/tables/hs_amt.c File Reference

```
#include "cfe.h"
#include "hs_tbl.h"
#include "hs_tbldefs.h"
#include "cfe_tbl_filedef.h"
```

Functions

- static [CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ \(\(__used__\)\)](#)

Variables

- [HS_AMTEntry_t HS_Default_AppMon_Tbl \[HS_MAX_MONITORED_APPS\]](#)

12.22.1 Detailed Description

The CFS Health and Safety (HS) Applications Monitor Table Definition

12.22.2 Function Documentation

12.22.2.1 __attribute__()

```
static CFE\_TBL\_FileDef\_t CFE\_TBL\_FileDef \_\_attribute\_\_ \(
    \_\_used\_\_) \[static\]
```

12.22.3 Variable Documentation

12.22.3.1 HS_Default_AppMon_Tbl

[HS_AMTEntry_t HS_Default_AppMon_Tbl \[HS_MAX_MONITORED_APPS\]](#)

Definition at line 37 of file hs_amt.c.

12.23 apps/hs/fsw/tables/hs_emt.c File Reference

```
#include "cfe.h"
#include "hs_tbl.h"
#include "hs_tbldefs.h"
#include "cfe_tbl_filedef.h"
```

Functions

- static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ ((__used__))

Variables

- HS_EMTEntry_t HS_Default_EventMon_Tbl [HS_MAX_MONITORED_EVENTS]

12.23.1 Detailed Description

The CFS Health and Safety (HS) Event Monitor Table Definition

12.23.2 Function Documentation

12.23.2.1 __attribute__()

```
static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ (
    __used__ ) [static]
```

12.23.3 Variable Documentation

12.23.3.1 HS_Default_EventMon_Tbl

`HS_EMTEntry_t HS_Default_EventMon_Tbl[HS_MAX_MONITORED_EVENTS]`

Initial value:

```
= {  
  
    {"CFE_ES", 0, 10, HS_EMT_ACT_NOACT},  
    {"CFE_EVS", 0, 10, HS_EMT_ACT_NOACT},  
    {"CFE_TIME", 0, 10, HS_EMT_ACT_NOACT},  
    {"CFE_TBL", 0, 10, HS_EMT_ACT_NOACT},  
    {"CFE_SBV", 0, 10, HS_EMT_ACT_NOACT},  
    {"", 0, 10, HS_EMT_ACT_NOACT},  
}
```

Definition at line 37 of file hs_emt.c.

12.24 apps/hs/fsw/tables/hs_mat.c File Reference

```
#include "cfe.h"
#include "hs_tbl.h"
#include "hs_tbldefs.h"
#include "cfe_tbl_filedef.h"
```

Functions

- static [CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ \(\(__used__\)\)](#)

Variables

- [HS_MATEEntry_t HS_Default_MsgActs_Tbl \[HS_MAX_MSG_ACT_TYPES\]](#)

12.24.1 Detailed Description

The CFS Health and Safety (HS) Message Actions Table Definition

12.24.2 Function Documentation

12.24.2.1 __attribute__()

```
static CFE\_TBL\_FileDef\_t CFE\_TBL\_FileDef \_\_attribute\_\_ \(\(\_\_used\_\_\)\) [static]
```

12.24.3 Variable Documentation

12.24.3.1 HS_Default_MsgActs_Tbl

[HS_MATEEntry_t HS_Default_MsgActs_Tbl \[HS_MAX_MSG_ACT_TYPES\]](#)

Definition at line 37 of file [hs_mat.c](#).

12.25 apps/hs/fsw/tables/hs_xct.c File Reference

```
#include "cfe.h"
#include "hs_tbl.h"
#include "hs_tbldefs.h"
#include "cfe_tbl_filedef.h"
```

Functions

- static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ ((__used__))

Variables

- HS_XCTEntry_t HS_Default_ExeCount_Tbl [HS_MAX_EXEC_CNT_SLOTS]

12.25.1 Detailed Description

The CFS Health and Safety (HS) Execution Counters Table Definition

12.25.2 Function Documentation

12.25.2.1 __attribute__()

```
static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ (
    __used__ ) [static]
```

12.25.3 Variable Documentation

12.25.3.1 HS_Default_ExeCount_Tbl

HS_XCTEntry_t HS_Default_ExeCount_Tbl [HS_MAX_EXEC_CNT_SLOTS]

Definition at line 37 of file hs_xct.c.

12.26 build/docs/osconfig-example.h File Reference

Macros

- #define **OS_MAX_TASKS**
Configuration file Operating System Abstraction Layer.
- #define **OS_MAX_QUEUES**
The maximum number of queues to support.
- #define **OS_MAX_COUNT_SEMAPHORES**
The maximum number of counting semaphores to support.
- #define **OS_MAX_BIN_SEMAPHORES**
The maximum number of binary semaphores to support.
- #define **OS_MAX_MUTEXES**
The maximum number of mutexes to support.
- #define **OS_MAX_MODULES**
The maximum number of modules to support.
- #define **OS_MAX_TIMEBASES**
The maximum number of timebases to support.
- #define **OS_MAX_TIMERS**
The maximum number of timer callbacks to support.
- #define **OS_MAX_NUM_OPEN_FILES**
The maximum number of concurrently open files to support.
- #define **OS_MAX_NUM_OPEN_DIRS**
The maximum number of concurrently open directories to support.
- #define **OS_MAX_FILE_SYSTEMS**
The maximum number of file systems to support.
- #define **OS_MAX_SYM_LEN**
The maximum length of symbols.
- #define **OS_MAX_FILE_NAME**
The maximum length of OSAL file names.
- #define **OS_MAX_PATH_LEN**
The maximum length of OSAL path names.
- #define **OS_MAX_API_NAME**
The maximum length of OSAL resource names.
- #define **OS SOCKADDR_MAX_LEN**
The maximum size of the socket address structure.
- #define **OS_BUFFER_SIZE**
The maximum size of output produced by a single `OS_printf()`.
- #define **OS_BUFFER_MSG_DEPTH**
The maximum number of `OS_printf()` output strings to buffer.
- #define **OS_UTILITYTASK_PRIORITY**
Priority level of the background utility task.
- #define **OS_UTILITYTASK_STACK_SIZE**
The stack size of the background utility task.
- #define **OS_MAX_CMD_LEN**
The maximum size of a shell command.
- #define **OS_QUEUE_MAX_DEPTH**

- `#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_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`

12.26.1 Macro Definition Documentation

12.26.1.1 OS_BUFFER_MSG_DEPTH

```
#define OS_BUFFER_MSG_DEPTH
```

The maximum number of [OS_printf\(\)](#) output strings to buffer.

Based on the OSAL_CONFIG_PRINTF_BUFFER_DEPTH configuration option

Definition at line 180 of file osconfig-example.h.

12.26.1.2 OS_BUFFER_SIZE

```
#define OS_BUFFER_SIZE
```

The maximum size of output produced by a single [OS_printf\(\)](#)

Based on the OSAL_CONFIG_PRINTF_BUFFER_SIZE configuration option

Definition at line 173 of file osconfig-example.h.

12.26.1.3 OS_FS_DEV_NAME_LEN

```
#define OS_FS_DEV_NAME_LEN 32
```

Device name length

Definition at line 265 of file osconfig-example.h.

12.26.1.4 OS_FS_PHYS_NAME_LEN

```
#define OS_FS_PHYS_NAME_LEN 64
```

Physical drive name length

Definition at line 266 of file osconfig-example.h.

12.26.1.5 OS_FS_VOL_NAME_LEN

```
#define OS_FS_VOL_NAME_LEN 32
```

Volume name length

Definition at line 267 of file osconfig-example.h.

12.26.1.6 OS_MAX_API_NAME

```
#define OS_MAX_API_NAME
```

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 156 of file osconfig-example.h.

Referenced by HS_MonitorEvent(), HS_ValidateAMTable(), and HS_ValidateEMTable().

12.26.1.7 OS_MAX_BIN_SEMAPHORES

```
#define OS_MAX_BIN_SEMAPHORES
```

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-example.h.

12.26.1.8 OS_MAX_CMD_LEN

```
#define OS_MAX_CMD_LEN
```

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 211 of file osconfig-example.h.

12.26.1.9 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 253 of file osconfig-example.h.

12.26.1.10 OS_MAX_COUNT_SEMAPHORES

```
#define OS_MAX_COUNT_SEMAPHORES
```

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-example.h.

12.26.1.11 OS_MAX_FILE_NAME

```
#define OS_MAX_FILE_NAME
```

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 135 of file osconfig-example.h.

12.26.1.12 OS_MAX_FILE_SYSTEMS

```
#define OS_MAX_FILE_SYSTEMS
```

The maximum number of file systems to support.

Based on the OSAL_CONFIG_MAX_FILE_SYSTEMS configuration option

Definition at line 114 of file osconfig-example.h.

12.26.1.13 OS_MAX_MODULES

```
#define OS_MAX_MODULES
```

The maximum number of modules to support.

Based on the OSAL_CONFIG_MAX_MODULES configuration option

Definition at line 79 of file osconfig-example.h.

12.26.1.14 OS_MAX_MUTEXES

```
#define OS_MAX_MUTEXES
```

The maximum number of mutexes to support.

Based on the OSAL_CONFIG_MAX_MUTEXES configuration option

Definition at line 72 of file osconfig-example.h.

12.26.1.15 OS_MAX_NUM_OPEN_DIRS

```
#define OS_MAX_NUM_OPEN_DIRS
```

The maximum number of concurrently open directories to support.

Based on the OSAL_CONFIG_MAX_NUM_OPEN_DIRS configuration option

Definition at line 107 of file osconfig-example.h.

12.26.1.16 OS_MAX_NUM_OPEN_FILES

```
#define OS_MAX_NUM_OPEN_FILES
```

The maximum number of concurrently open files to support.

Based on the OSAL_CONFIG_MAX_NUM_OPEN_FILES configuration option

Definition at line 100 of file osconfig-example.h.

12.26.1.17 OS_MAX_PATH_LEN

```
#define OS_MAX_PATH_LEN
```

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 147 of file osconfig-example.h.

12.26.1.18 OS_MAX_QUEUES

```
#define OS_MAX_QUEUES
```

The maximum number of queues to support.

Based on the OSAL_CONFIG_MAX_QUEUES configuration option

Definition at line 51 of file osconfig-example.h.

12.26.1.19 OS_MAX_SYM_LEN

```
#define OS_MAX_SYM_LEN
```

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 123 of file osconfig-example.h.

12.26.1.20 OS_MAX_TASKS

```
#define OS_MAX_TASKS
```

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/HS/HS/cfe/default_config.cmake. The maximum number of tasks to support

Based on the OSAL_CONFIG_MAX_TASKS configuration option

Definition at line 44 of file osconfig-example.h.

12.26.1.21 OS_MAX_TIMEBASES

```
#define OS_MAX_TIMEBASES
```

The maximum number of timebases to support.

Based on the OSAL_CONFIG_MAX_TIMEBASES configuration option

Definition at line 86 of file osconfig-example.h.

12.26.1.22 OS_MAX_TIMERS

```
#define OS_MAX_TIMERS
```

The maximum number of timer callbacks to support.

Based on the OSAL_CONFIG_MAX_TIMERS configuration option

Definition at line 93 of file osconfig-example.h.

12.26.1.23 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 260 of file osconfig-example.h.

12.26.1.24 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 238 of file osconfig-example.h.

12.26.1.25 OS_QUEUE_MAX_DEPTH

```
#define OS_QUEUE_MAX_DEPTH
```

The maximum depth of OSAL queues.

Based on the OSAL_CONFIG_QUEUE_MAX_DEPTH configuration option

Definition at line 218 of file osconfig-example.h.

12.26.1.26 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 228 of file osconfig-example.h.

12.26.1.27 OS_SOCKADDR_MAX_LEN

```
#define OS_SOCKADDR_MAX_LEN
```

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 166 of file osconfig-example.h.

12.26.1.28 OS_UTILITYTASK_PRIORITY

```
#define OS_UTILITYTASK_PRIORITY
```

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 190 of file osconfig-example.h.

12.26.1.29 OS_UTILITYTASK_STACK_SIZE

```
#define OS_UTILITYTASK_STACK_SIZE
```

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 200 of file osconfig-example.h.

12.27 cfe/cmake/sample_defs/cpu1_msgids.h File Reference

```
#include "cfe_mission_cfg.h"
```

Macros

- #define CFE_PLATFORM_CMD_MID_BASE 0x1800
Platform command message ID base offset.
- #define CFE_PLATFORM_TLM_MID_BASE 0x0800
Platform telemetry message ID base offset.
- #define CFE_PLATFORM_CMD_MID_BASE_GLOB 0x1860
"Global" command message ID base offset
- #define CFE_EVS_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_EVS_CMD_MSG /* 0x1801 */
- #define CFE_TEST_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TEST_CMD_MSG /* 0x1802 */
- #define CFE_SB_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_CMD_MSG /* 0x1803 */
- #define CFE_TBL_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TBL_CMD_MSG /* 0x1804 */
- #define CFE_TIME_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_CMD_MSG /* 0x1805 */
- #define CFE_ES_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_ES_CMD_MSG /* 0x1806 */
- #define CFE_ES_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_ES_SEND_HK_MSG /* 0x1808 */
- #define CFE_EVS_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_EVS_SEND_HK_MSG /* 0x1809 */
- #define CFE_SB_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_SEND_HK_MSG /* 0x180B */
- #define CFE_TBL_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TBL_SEND_HK_MSG /* 0x180C */
- #define CFE_TIME_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_SEND_HK_MSG /* 0x180D */
- #define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_SUB_RPT_CTRL_MSG /* 0x180E */
- #define CFE_TIME_TONE_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_TONE_CMD_MSG /* 0x1810 */
- #define CFE_TIME_1HZ_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_1HZ_CMD_MSG /* 0x1811 */
- #define CFE_TIME_DATA_CMD_MID CFE_PLATFORM_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_DATA_CMD_MSG /* 0x1860 */
- #define CFE_TIME_SEND_CMD_MID CFE_PLATFORM_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_SEND_CMD_MSG /* 0x1862 */
- #define CFE_ES_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_HK_TLM_MSG /* 0x0800 */
- #define CFE_EVS_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_HK_TLM_MSG /* 0x0801 */
- #define CFE_TEST_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TEST_HK_TLM_MSG /* 0x0802 */
- #define CFE_SB_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_HK_TLM_MSG /* 0x0803 */
- #define CFE_TBL_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TBL_HK_TLM_MSG /* 0x0804 */
- #define CFE_TIME_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TIME_HK_TLM_MSG /* 0x0805 */

- #define CFE_TIME_DIAG_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TIME_DIAG_TLMSG /* 0x0806 */
- #define CFE_ESV_LONG_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ESV_LONG_EVENT_MSG_MSG /* 0x0808 */
- #define CFE_ESV_SHORT_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ESV_SHORT_EVENT_MSG_MSG /* 0x0809 */
- #define CFE_SB_STATS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_STATS_TLMMSG /* 0x080A */
- #define CFE_ES_APP_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_APP_TLM_MSG /* 0x080B */
- #define CFE_TBL_REG_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TBL_REG_TLMMSG /* 0x080C */
- #define CFE_SB_ALLSUBS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_ALLSUBSTLM_MSG /* 0x080D */
- #define CFE_SB_ONESUB_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_ONESUB_TLM_MSG /* 0x080E */
- #define CFE_ES_MEMSTATS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_MEMSTATS_TLM_MSG /* 0x0810 */

12.27.1 Detailed Description

Purpose: This header file contains the Message Id's for messages used by the cFE core.

Author: R.McGraw/SSI

Notes: This file should not contain messages defined by cFE external applications.

12.27.2 Macro Definition Documentation

12.27.2.1 CFE_ES_APP_TLM_MID

```
#define CFE_ES_APP_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_APP_TLM_MSG /* 0x080B */
```

Definition at line 125 of file cpu1_msgids.h.

12.27.2.2 CFE_ES_CMD_MID

```
#define CFE_ES_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_ES_CMD_MSG /* 0x1806 */
```

Definition at line 92 of file cpu1_msgids.h.

12.27.2.3 CFE_ES_HK_TLM_MID

```
#define CFE_ES_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_HK_TLM_MSG /* 0x0800 */
```

Definition at line 115 of file cpu1_msgids.h.

12.27.2.4 CFE_ES_MEMSTATS_TLM_MID

```
#define CFE_ES_MEMSTATS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_ES_MEMSTATS_TLM_MSG /* 0x0810 */
```

Definition at line 129 of file cpu1_msgids.h.

12.27.2.5 CFE_ES_SEND_HK_MID

```
#define CFE_ES_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_ES_SEND_HK_MSG /* 0x1808 */
```

Definition at line 94 of file cpu1_msgids.h.

12.27.2.6 CFE_EVS_CMD_MID

```
#define CFE_EVS_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_EVS_CMD_MSG /* 0x1801 */
```

Definition at line 87 of file cpu1_msgids.h.

12.27.2.7 CFE_EVS_HK_TLM_MID

```
#define CFE_EVS_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_HK_TLM_MSG /* 0x0801 */
```

Definition at line 116 of file cpu1_msgids.h.

12.27.2.8 CFE_EVS_LONG_EVENT_MSG_MID

```
#define CFE_EVS_LONG_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_LONG_EVENT_MSG_MSG /* 0x0808 */
```

Definition at line 122 of file cpu1_msgids.h.

Referenced by HS_AcquirePointers(), HS_AppMain(), HS_DisableEventMonCmd(), and HS_EnableEventMonCmd().

12.27.2.9 CFE_EVS_SEND_HK_MID

```
#define CFE_EVS_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_EVS_SEND_HK_MSG /* 0x1809 */
```

Definition at line 95 of file cpu1_msgids.h.

12.27.2.10 CFE_EVS_SHORT_EVENT_MSG_MID

```
#define CFE_EVS_SHORT_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_SHORT_EVENT_MSG_ MSG /* 0x0809 */
```

Definition at line 123 of file cpu1_msgids.h.

Referenced by HS_AcquirePointers(), HS_AppMain(), HS_DisableEventMonCmd(), and HS_EnableEventMonCmd().

12.27.2.11 CFE_PLATFORM_CMD_MID_BASE

```
#define CFE_PLATFORM_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 59 of file cpu1_msgids.h.

12.27.2.12 CFE_PLATFORM_CMD_MID_BASE_GLOB

```
#define CFE_PLATFORM_CMD_MID_BASE_GLOB 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_PLATFORM_CMD_MID_BASE](#) for more information

Definition at line 82 of file cpu1_msgids.h.

12.27.2.13 CFE_PLATFORM_TLM_MID_BASE

```
#define CFE_PLATFORM_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_PLATFORM_CMD_MID_BASE](#) for more information

Definition at line 71 of file cpu1_msgids.h.

12.27.2.14 CFE_SB_ALLSUBS_TLM_MID

```
#define CFE_SB_ALLSUBS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_ALLSUBS_TLM_MSG /* 0x080←  
D */
```

Definition at line 127 of file cpu1_msgids.h.

12.27.2.15 CFE_SB_CMD_MID

```
#define CFE_SB_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_CMD_MSG /* 0x1803 */
```

Definition at line 89 of file cpu1_msgids.h.

12.27.2.16 CFE_SB_HK_TLM_MID

```
#define CFE_SB_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_HK_TLM_MSG /* 0x0803 */
```

Definition at line 118 of file cpu1_msgids.h.

12.27.2.17 CFE_SB_ONESUB_TLM_MID

```
#define CFE_SB_ONESUB_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_ONESUB_TLM_MSG /* 0x080E */
```

Definition at line 128 of file cpu1_msgids.h.

12.27.2.18 CFE_SB_SEND_HK_MID

```
#define CFE_SB_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_SEND_HK_MSG /* 0x180B */
```

Definition at line 97 of file cpu1_msgids.h.

12.27.2.19 CFE_SB_STATS_TLM_MID

```
#define CFE_SB_STATS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_STATS_TLM_MSG /* 0x080A */
```

Definition at line 124 of file cpu1_msgids.h.

12.27.2.20 CFE_SB_SUB_RPT_CTRL_MID

```
#define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_SUB_RPT_CTRL_MSG /* 0x180E */
```

Definition at line 101 of file cpu1_msgids.h.

12.27.2.21 CFE_TBL_CMD_MID

```
#define CFE_TBL_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TBL_CMD_MSG /* 0x1804 */
```

Definition at line 90 of file cpu1_msgids.h.

12.27.2.22 CFE_TBL_HK_TLM_MID

```
#define CFE_TBL_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TBL_HK_TLM_MSG /* 0x0804 */
```

Definition at line 119 of file cpu1_msgids.h.

12.27.2.23 CFE_TBL_REG_TLM_MID

```
#define CFE_TBL_REG_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TBL_REG_TLM_MSG /* 0x080C */
```

Definition at line 126 of file cpu1_msgids.h.

12.27.2.24 CFE_TBL_SEND_HK_MID

```
#define CFE_TBL_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TBL_SEND_HK_MSG /* 0x180C */
```

Definition at line 98 of file cpu1_msgids.h.

12.27.2.25 CFE_TEST_CMD_MID

```
#define CFE_TEST_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TEST_CMD_MSG /* 0x1802 */
```

Definition at line 88 of file cpu1_msgids.h.

12.27.2.26 CFE_TEST_HK_TLM_MID

```
#define CFE_TEST_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TEST_HK_TLM_MSG /* 0x0802 */
```

Definition at line 117 of file cpu1_msgids.h.

12.27.2.27 CFE_TIME_1HZ_CMD_MID

```
#define CFE_TIME_1HZ_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_1HZ_CMD_MSG /* 0x1811 */
```

Definition at line 104 of file cpu1_msgids.h.

12.27.2.28 CFE_TIME_CMD_MID

```
#define CFE_TIME_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_CMD_MSG /* 0x1805 */
```

Definition at line 91 of file cpu1_msgids.h.

12.27.2.29 CFE_TIME_DATA_CMD_MID

```
#define CFE_TIME_DATA_CMD_MID CFE_PLATFORM_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_DATA_CMD_MSG /* 0x1860 */
```

Definition at line 109 of file cpu1_msgids.h.

12.27.2.30 CFE_TIME_DIAG_TLM_MID

```
#define CFE_TIME_DIAG_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TIME_DIAG_TLM_MSG /* 0x0806 */
```

Definition at line 121 of file cpu1_msgids.h.

12.27.2.31 CFE_TIME_HK_TLM_MID

```
#define CFE_TIME_HK_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TIME_HK_TLM_MSG /* 0x0805 */
```

Definition at line 120 of file cpu1_msgids.h.

12.27.2.32 CFE_TIME_SEND_CMD_MID

```
#define CFE_TIME_SEND_CMD_MID CFE_PLATFORM_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_SEND_CMD_MSG /* 0x1862 */
```

Definition at line 110 of file cpu1_msgids.h.

12.27.2.33 CFE_TIME_SEND_HK_MID

```
#define CFE_TIME_SEND_HK_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_SEND_HK_MSG /* 0x180D */
```

Definition at line 99 of file cpu1_msgids.h.

12.27.2.34 CFE_TIME_TONE_CMD_MID

```
#define CFE_TIME_TONE_CMD_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_TIME_TONE_CMD_MSG /* 0x1810 */
```

Definition at line 103 of file cpu1_msgids.h.

12.28 cfe/cmake/sample_defs/cpu1_platform_cfg.h File Reference

Macros

- #define CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING "/cf"
- #define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"
- #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_ENDIAN CCSDS_LITTLE_ENDIAN
- #define CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME "/ram/cfe_sb_route.dat"
- #define CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME "/ram/cfe_sb_pipe.dat"
- #define CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME "/ram/cfe_sb_msgmap.dat"
- #define CFE_PLATFORM_SB_FILTERED_EVENT1 CFE_SB_SEND_NO_SUBS_EID
- #define CFE_PLATFORM_SB_FILTER_MASK1 CFE_EVS_FIRST_4_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
- #define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
- #define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
- #define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT5 0
- #define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT6 0
- #define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT7 0
- #define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT8 0
- #define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01 8
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
- #define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_MSG_SIZE + 128)
- #define CFE_PLATFORM_TIME_CFG_SERVER true

- #define CFE_PLATFORM_TIME_CFG_CLIENT false
- #define CFE_PLATFORM_TIME_CFG_VIRTUAL true
- #define CFE_PLATFORM_TIME_CFG_SIGNAL false
- #define CFE_PLATFORM_TIME_CFG_SOURCE false
- #define CFE_PLATFORM_TIME_CFG_SRC_MET false
- #define CFE_PLATFORM_TIME_CFG_SRC_GPS false
- #define CFE_PLATFORM_TIME_CFG_SRC_TIME false
- #define CFE_PLATFORM_TIME_MAX_DELTA_SECS 0
- #define CFE_PLATFORM_TIME_MAX_DELTA_SUBS 500000
- #define CFE_PLATFORM_TIME_MAX_LOCAL_SECS 27
- #define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
- #define CFE_PLATFORM_TIME_CFG_TONE_LIMIT 20000
- #define CFE_PLATFORM_TIME_CFG_START_FLY 2
- #define CFE_PLATFORM_TIME_CFG_LATCH_FLY 8
- #define CFE_PLATFORM_ES_MAX_APPLICATIONS 32
- #define CFE_PLATFORM_ES_MAX_LIBRARIES 10
- #define CFE_PLATFORM_ES_ER_LOG_ENTRIES 20
- #define CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE 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_RESET_AREA_SIZE (170 * 1024)
- #define CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN 4
- #define CFE_PLATFORM_ES_NONVOL_STARTUP_FILE "/cf/cfe_es_startup.scr"
- #define CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE "/ram/cfe_es_startup.scr"
- #define CFE_PLATFORM_ES_DEFAULT_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_EVS_START_TASK_PRIORITY 61
- #define CFE_PLATFORM_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
- #define CFE_PLATFORM_SB_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_ES_START_TASK_PRIORITY 68
- #define CFE_PLATFORM_ES_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_TIME_START_TASK_PRIORITY 60
- #define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
- #define CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE 8192
- #define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
- #define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES 512
- #define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
- #define CFE_PLATFORM_ES_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_EVS_MAX_EVENT_FILTERS 8
- #define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE "/ram/cfe_evs.log"
- #define CFE_PLATFORM_EVS_LOG_MAX 20
- #define CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE "/ram/cfe_evs_app.dat"
- #define CFE_PLATFORM_EVS_PORT_DEFAULT 0x0001
- #define CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG 0xE
- #define CFE_PLATFORM_EVS_DEFAULT_LOG_MODE 1
- #define CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE CFE_EVS_MsgFormat_LONG
- #define CFE_PLATFORM_TBL_BUF_MEMORY_BYTES 524288
- #define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE 16384
- #define CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE 16384
- #define CFE_PLATFORM_TBL_MAX_NUM_TABLES 128
- #define CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES 32
- #define CFE_PLATFORM_TBL_MAX_NUM_HANDLES 256
- #define CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS 4
- #define CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS 10
- #define CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE "/ram/cfe_tbl_reg.log"
- #define CFE_PLATFORM_TBL_VALID_SCID_COUNT 0
- #define CFE_PLATFORM_TBL_U32FROM4CHARS(_C1, _C2, _C3, _C4) ((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_ES_STARTUP_SYNC_POLL_MSEC 50
- #define CFE_PLATFORM_CORE_MAX_STARTUP_MSEC 30000
- #define CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC 1000

12.28.1 Detailed Description

Purpose: This header file contains the platform configuration parameters.

Notes: The impact of changing these configurations from their default value is not yet documented. Changing these values may impact the performance and functionality of the system.

Author: R.McGraw/SSI

12.28.2 Macro Definition Documentation

12.28.2.1 CFE_PLATFORM_CORE_MAX_STARTUP_MSEC

```
#define CFE_PLATFORM_CORE_MAX_STARTUP_MSEC 30000
```

Purpose CFE core application startup timeout

Description:

The upper limit for the amount of time that the cFE core applications (ES, SB, EVS, TIME, TBL) are each 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 1689 of file cpu1_platform_cfg.h.

12.28.2.2 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 190 of file cpu1_platform_cfg.h.

12.28.2.3 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 620 of file cpu1_platform_cfg.h.

12.28.2.4 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 591 of file cpu1_platform_cfg.h.

12.28.2.5 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE 80000
```

Definition at line 1325 of file cpu1_platform_cfg.h.

12.28.2.6 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 1215 of file cpu1_platform_cfg.h.

12.28.2.7 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 1309 of file cpu1_platform_cfg.h.

12.28.2.8 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 16
```

Definition at line 1310 of file cpu1_platform_cfg.h.

12.28.2.9 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 32
```

Definition at line 1311 of file cpu1_platform_cfg.h.

12.28.2.10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 48
```

Definition at line 1312 of file cpu1_platform_cfg.h.

12.28.2.11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 64
```

Definition at line 1313 of file cpu1_platform_cfg.h.

12.28.2.12 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 96
```

Definition at line 1314 of file cpu1_platform_cfg.h.

12.28.2.13 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 128
```

Definition at line 1315 of file cpu1_platform_cfg.h.

12.28.2.14 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 160
```

Definition at line 1316 of file cpu1_platform_cfg.h.

12.28.2.15 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 256
```

Definition at line 1317 of file cpu1_platform_cfg.h.

12.28.2.16 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 512
```

Definition at line 1318 of file cpu1_platform_cfg.h.

12.28.2.17 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 1024
```

Definition at line 1319 of file cpu1_platform_cfg.h.

12.28.2.18 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 2048
```

Definition at line 1320 of file cpu1_platform_cfg.h.

12.28.2.19 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 4096
```

Definition at line 1321 of file cpu1_platform_cfg.h.

12.28.2.20 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 8192
```

Definition at line 1322 of file cpu1_platform_cfg.h.

12.28.2.21 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 16384
```

Definition at line 1323 of file cpu1_platform_cfg.h.

12.28.2.22 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 32768
```

Definition at line 1324 of file cpu1_platform_cfg.h.

12.28.2.23 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 697 of file cpu1_platform_cfg.h.

12.28.2.24 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 802 of file cpu1_platform_cfg.h.

12.28.2.25 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 876 of file cpu1_platform_cfg.h.

12.28.2.26 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 848 of file cpu1_platform_cfg.h.

12.28.2.27 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 862 of file cpu1_platform_cfg.h.

12.28.2.28 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 894 of file cpu1_platform_cfg.h.

12.28.2.29 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 912 of file cpu1_platform_cfg.h.

12.28.2.30 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 1062 of file cpu1_platform_cfg.h.

12.28.2.31 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 833 of file `cpu1_platform_cfg.h`.

12.28.2.32 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 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 817 of file `cpu1_platform_cfg.h`.

12.28.2.33 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 518 of file cpu1_platform_cfg.h.

12.28.2.34 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 532 of file cpu1_platform_cfg.h.

12.28.2.35 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 491 of file cpu1_platform_cfg.h.

12.28.2.36 CFE_PLATFORM_ES_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000
```

Definition at line 1298 of file cpu1_platform_cfg.h.

12.28.2.37 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 572 of file cpu1_platform_cfg.h.

12.28.2.38 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 505 of file cpu1_platform_cfg.h.

12.28.2.39 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 1263 of file cpu1_platform_cfg.h.

12.28.2.40 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 1230 of file cpu1_platform_cfg.h.

12.28.2.41 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 1282 of file cpu1_platform_cfg.h.

12.28.2.42 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 16
```

Definition at line 1283 of file cpu1_platform_cfg.h.

12.28.2.43 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 32
```

Definition at line 1284 of file cpu1_platform_cfg.h.

12.28.2.44 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 48
```

Definition at line 1285 of file cpu1_platform_cfg.h.

12.28.2.45 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 64
```

Definition at line 1286 of file cpu1_platform_cfg.h.

12.28.2.46 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 96
```

Definition at line 1287 of file cpu1_platform_cfg.h.

12.28.2.47 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 128
```

Definition at line 1288 of file cpu1_platform_cfg.h.

12.28.2.48 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 160
```

Definition at line 1289 of file cpu1_platform_cfg.h.

12.28.2.49 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 256
```

Definition at line 1290 of file cpu1_platform_cfg.h.

12.28.2.50 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 512
```

Definition at line 1291 of file cpu1_platform_cfg.h.

12.28.2.51 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 1024
```

Definition at line 1292 of file cpu1_platform_cfg.h.

12.28.2.52 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 2048
```

Definition at line 1293 of file cpu1_platform_cfg.h.

12.28.2.53 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 4096
```

Definition at line 1294 of file cpu1_platform_cfg.h.

12.28.2.54 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 8192
```

Definition at line 1295 of file cpu1_platform_cfg.h.

12.28.2.55 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 16384
```

Definition at line 1296 of file cpu1_platform_cfg.h.

12.28.2.56 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 32768
```

Definition at line 1297 of file cpu1_platform_cfg.h.

12.28.2.57 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 759 of file cpu1_platform_cfg.h.

12.28.2.58 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 47 of file cpu1_platform_cfg.h.

12.28.2.59 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 773 of file cpu1_platform_cfg.h.

12.28.2.60 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 561 of file cpu1_platform_cfg.h.

12.28.2.61 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 1036 of file cpu1_platform_cfg.h.

12.28.2.62 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 1007 of file cpu1_platform_cfg.h.

12.28.2.63 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 1021 of file cpu1_platform_cfg.h.

12.28.2.64 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 928 of file cpu1_platform_cfg.h.

12.28.2.65 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 1046 of file cpu1_platform_cfg.h.

12.28.2.66 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 948 of file cpu1_platform_cfg.h.

12.28.2.67 CFE_PLATFORM_ES_PERF_FILTMASK_INIT

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_ALL
```

Purpose Define Default Filter Mask Setting for Performance Data Buffer

Description:

Defines the default filter mask for the performance data buffer. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 959 of file cpu1_platform_cfg.h.

12.28.2.68 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 938 of file cpu1_platform_cfg.h.

12.28.2.69 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_ALL ~CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

Purpose Define Filter Trigger Setting for Enabling All Performance Entries

Description:

Defines the trigger mask for enabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 981 of file cpu1_platform_cfg.h.

12.28.2.70 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_INIT CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

Purpose Define Default Filter Trigger Setting for Performance Data Buffer

Description:

Defines the default trigger mask for the performance data buffer. The value is a 32-bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 992 of file cpu1_platform_cfg.h.

12.28.2.71 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 970 of file cpu1_platform_cfg.h.

12.28.2.72 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 1245 of file cpu1_platform_cfg.h.

12.28.2.73 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 63 of file cpu1_platform_cfg.h.

12.28.2.74 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 656 of file cpu1_platform_cfg.h.

12.28.2.75 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 680 of file `cpu1_platform_cfg.h`.

12.28.2.76 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 638 of file `cpu1_platform_cfg.h`.

12.28.2.77 CFE_PLATFORM_ES_RESET_AREA_SIZE

```
#define CFE_PLATFORM_ES_RESET_AREA_SIZE (170 * 1024)
```

Purpose Define ES Reset Area Size

Description:

The ES Reset Area Size. This is the size in bytes of the cFE Reset variable and log area. This is a block of memory used by the cFE to store the system log ER Log and critical reset variables. This is 4 of 4 of the memory areas that are preserved during a processor reset. Note: This area must be sized large enough to hold all of the data structures. It should be automatically sized based on the CFE_ES_ResetData_t type, but circular dependencies in the headers prevent it from being defined this way. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 153600 (150KBytes) and an upper limit of UINT_MAX (4 Gigabytes) on this configuration parameter.

Definition at line 740 of file cpu1_platform_cfg.h.

12.28.2.78 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 1125 of file cpu1_platform_cfg.h.

12.28.2.79 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 1140 of file cpu1_platform_cfg.h.

12.28.2.80 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 1707 of file cpu1_platform_cfg.h.

12.28.2.81 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 1665 of file cpu1_platform_cfg.h.

12.28.2.82 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 547 of file cpu1_platform_cfg.h.

12.28.2.83 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 717 of file `cpu1_platform_cfg.h`.

12.28.2.84 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 787 of file `cpu1_platform_cfg.h`.

12.28.2.85 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 1379 of file cpu1_platform_cfg.h.

12.28.2.86 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 1352 of file cpu1_platform_cfg.h.

12.28.2.87 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 1426 of file cpu1_platform_cfg.h.

12.28.2.88 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 1439 of file `cpu1_platform_cfg.h`.

12.28.2.89 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG

```
#define CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG 0xE
```

Purpose Default EVS Event Type Filter Mask

Description:

Defines a state of on or off for all four event types. The term event 'type' refers to the criticality level and may be Debug, Informational, Error or Critical. Each event type has a bit position. (bit 0 = Debug, bit 1 = Info, bit 2 = Error, bit 3 = Critical). This is a global setting, meaning it applies to all applications. To filter an event type, set its bit to zero. For example, 0xE means Debug = OFF, Info = ON, Error = ON, Critical = ON

Limits

The valid settings are 0x0 to 0xF.

Definition at line 1410 of file `cpu1_platform_cfg.h`.

12.28.2.90 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 1364 of file `cpu1_platform_cfg.h`.

12.28.2.91 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 1338 of file cpu1_platform_cfg.h.

12.28.2.92 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 1393 of file cpu1_platform_cfg.h.

12.28.2.93 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 1073 of file cpu1_platform_cfg.h.

12.28.2.94 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 1088 of file cpu1_platform_cfg.h.

12.28.2.95 CFE_PLATFORM_SB_BUF_MEMORY_BYTES

```
#define CFE_PLATFORM_SB_BUF_MEMORY_BYTES 524288
```

Purpose Size of the SB buffer memory pool

Description:

Dictates the size of the SB memory pool. For each message the SB sends, the SB dynamically allocates from this memory pool, the memory needed to process the message. The memory needed to process each message is msg size + msg descriptor(CFE_SB_BufferD_t). This memory pool is also used to allocate destination descriptors (CFE_SB_DestinationD_t) during the subscription process. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'. Some memory statistics have been added to the SB housekeeping packet. NOTE: It is important to monitor these statistics to ensure the desired memory margin is met.

Limits

This parameter has a lower limit of 512 and an upper limit of `UINT_MAX` (4 Gigabytes).

Definition at line 153 of file cpu1_platform_cfg.h.

12.28.2.96 CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME

```
#define CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME "/ram/cfe_sb_msgmap.dat"
```

Purpose Default Message Map Filename

Description:

The value of this constant defines the filename used to store the software bus message map information. This filename is used only when no filename is specified in the command. The message map is a lookup table (array of 16bit words) that has an element for each possible MsgId value and holds the routing table index for that MsgId. The Msg Map provides fast access to the destinations of a message.

Limits

The length of each string, including the NULL terminator cannot exceed the [OS_MAX_PATH_LEN](#) value.

Definition at line 235 of file cpu1_platform_cfg.h.

12.28.2.97 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 131 of file cpu1_platform_cfg.h.

12.28.2.98 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME

```
#define CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME "/ram/cfe_sb_pipe.dat"
```

Purpose Default Pipe Information Filename

Description:

The value of this constant defines the filename used to store the software bus pipe information. This filename is used only when no filename is specified in the command.

Limits

The length of each string, including the NULL terminator cannot exceed the [OS_MAX_PATH_LEN](#) value.

Definition at line 218 of file cpu1_platform_cfg.h.

12.28.2.99 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 204 of file cpu1_platform_cfg.h.

12.28.2.100 CFE_PLATFORM_SB_FILTER_MASK1

```
#define CFE_PLATFORM_SB_FILTER_MASK1 CFE_EVS_FIRST_4_STOP
```

Definition at line 253 of file cpu1_platform_cfg.h.

12.28.2.101 CFE_PLATFORM_SB_FILTER_MASK2

```
#define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
```

Definition at line 256 of file cpu1_platform_cfg.h.

12.28.2.102 CFE_PLATFORM_SB_FILTER_MASK3

```
#define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
```

Definition at line 259 of file cpu1_platform_cfg.h.

12.28.2.103 CFE_PLATFORM_SB_FILTER_MASK4

```
#define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
```

Definition at line 262 of file cpu1_platform_cfg.h.

12.28.2.104 CFE_PLATFORM_SB_FILTER_MASK5

```
#define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
```

Definition at line 265 of file cpu1_platform_cfg.h.

12.28.2.105 CFE_PLATFORM_SB_FILTER_MASK6

```
#define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
```

Definition at line 268 of file cpu1_platform_cfg.h.

12.28.2.106 CFE_PLATFORM_SB_FILTER_MASK7

```
#define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
```

Definition at line 271 of file cpu1_platform_cfg.h.

12.28.2.107 CFE_PLATFORM_SB_FILTER_MASK8

```
#define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
```

Definition at line 274 of file cpu1_platform_cfg.h.

12.28.2.108 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 EVS. The filtering will begin after the SB task initializes and stay in effect until a cmd to EVS changes it. This allows the operator to set limits on the number of event messages that are sent during system initialization. NOTE: Set all unused event values and mask values to zero

Limits

This filtering applies only to SB events. These parameters have a lower limit of 0 and an upper limit of 65535.

Definition at line 252 of file cpu1_platform_cfg.h.

12.28.2.109 CFE_PLATFORM_SB_FILTERED_EVENT2

```
#define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
```

Definition at line 255 of file cpu1_platform_cfg.h.

12.28.2.110 CFE_PLATFORM_SB_FILTERED_EVENT3

```
#define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
```

Definition at line 258 of file cpu1_platform_cfg.h.

12.28.2.111 CFE_PLATFORM_SB_FILTERED_EVENT4

```
#define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
```

Definition at line 261 of file cpu1_platform_cfg.h.

12.28.2.112 CFE_PLATFORM_SB_FILTERED_EVENT5

```
#define CFE_PLATFORM_SB_FILTERED_EVENT5 0
```

Definition at line 264 of file cpu1_platform_cfg.h.

12.28.2.113 CFE_PLATFORM_SB_FILTERED_EVENT6

```
#define CFE_PLATFORM_SB_FILTERED_EVENT6 0
```

Definition at line 267 of file cpu1_platform_cfg.h.

12.28.2.114 CFE_PLATFORM_SB_FILTERED_EVENT7

```
#define CFE_PLATFORM_SB_FILTERED_EVENT7 0
```

Definition at line 270 of file cpu1_platform_cfg.h.

12.28.2.115 CFE_PLATFORM_SB_FILTERED_EVENT8

```
#define CFE_PLATFORM_SB_FILTERED_EVENT8 0
```

Definition at line 273 of file cpu1_platform_cfg.h.

12.28.2.116 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

CFE_SB_INVALID_MSG is set to the maximum representable number of type [CFE_SB_MsgId_t](#). CFE_PLATFORM_SB_HIGHEST_VALID_MSGID lower limit is 1, up to CFE_SB_INVALID_MSG_ID - 1.

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 179 of file cpu1_platform_cfg.h.

12.28.2.117 CFE_PLATFORM_SB_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_MSG_SIZE + 128)
```

Definition at line 303 of file cpu1_platform_cfg.h.

12.28.2.118 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 116 of file cpu1_platform_cfg.h.

12.28.2.119 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 83 of file cpu1_platform_cfg.h.

12.28.2.120 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 100 of file cpu1_platform_cfg.h.

12.28.2.121 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 287 of file cpu1_platform_cfg.h.

12.28.2.122 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16
```

Definition at line 288 of file cpu1_platform_cfg.h.

12.28.2.123 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20
```

Definition at line 289 of file cpu1_platform_cfg.h.

12.28.2.124 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36
```

Definition at line 290 of file cpu1_platform_cfg.h.

12.28.2.125 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64
```

Definition at line 291 of file cpu1_platform_cfg.h.

12.28.2.126 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96
```

Definition at line 292 of file cpu1_platform_cfg.h.

12.28.2.127 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
```

Definition at line 293 of file cpu1_platform_cfg.h.

12.28.2.128 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
```

Definition at line 294 of file cpu1_platform_cfg.h.

12.28.2.129 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
```

Definition at line 295 of file cpu1_platform_cfg.h.

12.28.2.130 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
```

Definition at line 296 of file cpu1_platform_cfg.h.

12.28.2.131 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
```

Definition at line 297 of file cpu1_platform_cfg.h.

12.28.2.132 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
```

Definition at line 298 of file cpu1_platform_cfg.h.

12.28.2.133 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
```

Definition at line 299 of file cpu1_platform_cfg.h.

12.28.2.134 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
```

Definition at line 300 of file cpu1_platform_cfg.h.

12.28.2.135 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
```

Definition at line 301 of file cpu1_platform_cfg.h.

12.28.2.136 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
```

Definition at line 302 of file cpu1_platform_cfg.h.

12.28.2.137 CFE_PLATFORM_SB_START_TASK_PRIORITY

```
#define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
```

Purpose Define SB Task Priority

Description:

Defines the cFE_SB Task priority.

Limits

Not Applicable

Definition at line 1099 of file cpu1_platform_cfg.h.

12.28.2.138 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 1114 of file cpu1_platform_cfg.h.

12.28.2.139 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 1455 of file cpu1_platform_cfg.h.

12.28.2.140 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 1569 of file cpu1_platform_cfg.h.

12.28.2.141 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 1510 of file cpu1_platform_cfg.h.

12.28.2.142 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_BUFSIZE](#).

Definition at line 1467 of file cpu1_platform_cfg.h.

12.28.2.143 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_MAX_NUM_TABLES](#)) and should be set higher if tables are shared between applications.

Definition at line 1523 of file cpu1_platform_cfg.h.

12.28.2.144 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 1496 of file cpu1_platform_cfg.h.

12.28.2.145 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 1556 of file cpu1_platform_cfg.h.

12.28.2.146 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 1538 of file cpu1_platform_cfg.h.

12.28.2.147 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_BYTES](#).

Definition at line 1483 of file cpu1_platform_cfg.h.

12.28.2.148 CFE_PLATFORM_TBL_START_TASK_PRIORITY

```
#define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
```

Purpose Define TBL Task Priority

Description:

Defines the cFE_TBL Task priority.

Limits

Not Applicable

Definition at line 1187 of file cpu1_platform_cfg.h.

12.28.2.149 CFE_PLATFORM_TBL_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define TBL Task Stack Size

Description:

Defines the cFE_TBL Task Stack Size

Limits

There is a lower limit of 2048 on this configuration 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 1202 of file cpu1_platform_cfg.h.

12.28.2.150 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 1591 of file cpu1_platform_cfg.h.

12.28.2.151 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 1640 of file cpu1_platform_cfg.h.

12.28.2.152 CFE_PLATFORM_TBL_VALID_PRID_2

```
#define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
```

Definition at line 1641 of file cpu1_platform_cfg.h.

12.28.2.153 CFE_PLATFORM_TBL_VALID_PRID_3

```
#define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

Definition at line 1642 of file cpu1_platform_cfg.h.

12.28.2.154 CFE_PLATFORM_TBL_VALID_PRID_4

```
#define CFE_PLATFORM_TBL_VALID_PRID_4 0
```

Definition at line 1643 of file cpu1_platform_cfg.h.

12.28.2.155 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 1626 of file cpu1_platform_cfg.h.

12.28.2.156 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 1606 of file cpu1_platform_cfg.h.

12.28.2.157 CFE_PLATFORM_TBL_VALID_SCID_2

```
#define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
```

Definition at line 1607 of file cpu1_platform_cfg.h.

12.28.2.158 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 1588 of file cpu1_platform_cfg.h.

12.28.2.159 CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY 25
```

Definition at line 1157 of file cpu1_platform_cfg.h.

12.28.2.160 CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE 8192
```

Definition at line 1176 of file cpu1_platform_cfg.h.

12.28.2.161 CFE_PLATFORM_TIME_CFG_CLIENT

```
#define CFE_PLATFORM_TIME_CFG_CLIENT false
```

Definition at line 318 of file cpu1_platform_cfg.h.

12.28.2.162 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 475 of file cpu1_platform_cfg.h.

12.28.2.163 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 317 of file cpu1_platform_cfg.h.

12.28.2.164 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 365 of file cpu1_platform_cfg.h.

12.28.2.165 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 385 of file [cpu1_platform_cfg.h](#).

12.28.2.166 CFE_PLATFORM_TIME_CFG_SRC_GPS

```
#define CFE_PLATFORM_TIME_CFG_SRC_GPS false
```

Definition at line 402 of file [cpu1_platform_cfg.h](#).

12.28.2.167 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 401 of file [cpu1_platform_cfg.h](#).

12.28.2.168 CFE_PLATFORM_TIME_CFG_SRC_TIME

```
#define CFE_PLATFORM_TIME_CFG_SRC_TIME false
```

Definition at line 403 of file cpu1_platform_cfg.h.

12.28.2.169 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 462 of file cpu1_platform_cfg.h.

12.28.2.170 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 450 of file cpu1_platform_cfg.h.

12.28.2.171 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 350 of file cpu1_platform_cfg.h.

12.28.2.172 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 422 of file cpu1_platform_cfg.h.

12.28.2.173 CFE_PLATFORM_TIME_MAX_DELTA_SUBS

```
#define CFE_PLATFORM_TIME_MAX_DELTA_SUBS 500000
```

Definition at line 423 of file cpu1_platform_cfg.h.

12.28.2.174 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 435 of file cpu1_platform_cfg.h.

12.28.2.175 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS

```
#define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
```

Definition at line 436 of file cpu1_platform_cfg.h.

12.28.2.176 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 1155 of file cpu1_platform_cfg.h.

12.28.2.177 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 1174 of file cpu1_platform_cfg.h.

12.28.2.178 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
```

Definition at line 1156 of file cpu1_platform_cfg.h.

12.28.2.179 CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
```

Definition at line 1175 of file cpu1_platform_cfg.h.

12.29 cfe/cmake/sample_defs/sample_mission_cfg.h File Reference

Macros

- #define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768
- #define CFE_MISSION_TIME_CFG_DEFAULT_TAI true
- #define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
- #define CFE_MISSION_TIME_CFG_FAKE_TONE true
- #define CFE_MISSION_TIME_AT_TONE_WAS true
- #define CFE_MISSION_TIME_AT_TONE_WILL_BE false
- #define CFE_MISSION_TIME_MIN_ELAPSED 0
- #define CFE_MISSION_TIME_MAX_ELAPSED 200000
- #define CFE_MISSION_TIME_DEF_MET_SECS 1000

- #define CFE_MISSION_TIME_DEF_MET_SUBS 0
- #define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
- #define CFE_MISSION_TIME_DEF_STCF_SUBS 0
- #define CFE_MISSION_TIME_DEF_LEAPS 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
- #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
- #define CFE_MISSION_ES_MAX_MESSAGE_LENGTH 122
- #define CFE_MISSION_ES_DEFAULT_CRC CFE_MISSION_ES_CRC_16
- #define CFE_MISSION_TBL_MAX_NAME_LENGTH 16
- #define CFE_MISSION_ES_CMD_MSG 1
- #define CFE_MISSION_TEST_CMD_MSG 2
- #define CFE_MISSION_SB_CMD_MSG 3
- #define CFE_MISSION_TBL_CMD_MSG 4
- #define CFE_MISSION_TIME_CMD_MSG 5
- #define CFE_MISSION_ES_CMD_MSG 6
- #define CFE_MISSION_ES_SEND_HK_MSG 8
- #define CFE_MISSION_ES_SEND_HK_MSG 9
- #define CFE_MISSION_SB_SEND_HK_MSG 11
- #define CFE_MISSION_TBL_SEND_HK_MSG 12
- #define CFE_MISSION_TIME_SEND_HK_MSG 13
- #define CFE_MISSION_SB_SUB_RPT_CTRL_MSG 14
- #define CFE_MISSION_TIME_TONE_CMD_MSG 16
- #define CFE_MISSION_TIME_1HZ_CMD_MSG 17
- #define CFE_MISSION_TIME_DATA_CMD_MSG 0
- #define CFE_MISSION_TIME_SEND_CMD_MSG 2
- #define CFE_MISSION_ES_HK_TLM_MSG 0
- #define CFE_MISSION_ES_HK_TLM_MSG 1
- #define CFE_MISSION_TEST_HK_TLM_MSG 2
- #define CFE_MISSION_SB_HK_TLM_MSG 3
- #define CFE_MISSION_TBL_HK_TLM_MSG 4
- #define CFE_MISSION_TIME_HK_TLM_MSG 5
- #define CFE_MISSION_TIME_DIAG_TLM_MSG 6
- #define CFE_MISSION_ES_LONG_EVENT_MSG_MSG 8
- #define CFE_MISSION_ES_SHORT_EVENT_MSG_MSG 9
- #define CFE_MISSION_SB_STATS_TLM_MSG 10
- #define CFE_MISSION_ES_APP_TLM_MSG 11
- #define CFE_MISSION_TBL_REG_TLM_MSG 12
- #define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
- #define CFE_MISSION_SB_ONESUB_TLM_MSG 14
- #define CFE_MISSION_ES_MEMSTATS_TLM_MSG 16
- #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_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)
- #define CFE_MISSION_SB_MAX_PIPES 64
- #define CFE_MISSION_MAX_PATH_LEN 64
- #define CFE_MISSION_MAX_FILE_LEN 20
- #define CFE_MISSION_MAX_API_LEN 20
- #define CFE_MISSION_ES_CDS_MAX_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 1
CRC (8 bit additive - returns 32 bit total) (Currently not implemented)
- #define CFE_MISSION_ES_CRC_16 2
CRC (16 bit additive - returns 32 bit total)
- #define CFE_MISSION_ES_CRC_32 3
CRC (32 bit additive - returns 32 bit total) (Currently not implemented) .

12.29.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

Notes: The impact of changing these configurations from their default value is not yet documented. Changing these values may impact the performance and functionality of the system.

Author: R.McGraw/SSI

12.29.2 Macro Definition Documentation

12.29.2.1 CFE_MISSION_ES_APP_TLM_MSG

```
#define CFE_MISSION_ES_APP_TLM_MSG 11
```

Definition at line 366 of file sample_mission_cfg.h.

12.29.2.2 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_M->  
AX_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.C->DSName"

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 561 of file sample_mission_cfg.h.

12.29.2.3 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 245 of file sample_mission_cfg.h.

12.29.2.4 CFE_MISSION_ES_CMD_MSG

```
#define CFE_MISSION_ES_CMD_MSG 6
```

Definition at line 316 of file sample_mission_cfg.h.

12.29.2.5 CFE_MISSION_ES_CRC_16

```
#define CFE_MISSION_ES_CRC_16 2
```

CRC (16 bit additive - returns 32 bit total)

Definition at line 264 of file sample_mission_cfg.h.

12.29.2.6 CFE_MISSION_ES_CRC_32

```
#define CFE_MISSION_ES_CRC_32 3
```

CRC (32 bit additive - returns 32 bit total) (Currently not implemented) .

Definition at line 265 of file sample_mission_cfg.h.

12.29.2.7 CFE_MISSION_ES_CRC_8

```
#define CFE_MISSION_ES_CRC_8 1
```

CRC (8 bit additive - returns 32 bit total) (Currently not implemented)

Definition at line 263 of file sample_mission_cfg.h.

12.29.2.8 CFE_MISSION_ES_DEFAULT_CRC

```
#define CFE_MISSION_ES_DEFAULT_CRC CFE_MISSION_ES_CRC_16
```

Purpose Mission Default CRC algorithm

Description:

Indicates the which CRC algorithm should be used as the default for verifying the contents of Critical Data Stores and when calculating Table Image data integrity values.

Limits

Currently only CFE_MISSION_ES_CRC_16 is supported (see [CFE_MISSION_ES_CRC_16](#))

Definition at line 282 of file sample_mission_cfg.h.

12.29.2.9 CFE_MISSION_ES_HK_TLM_MSG

```
#define CFE_MISSION_ES_HK_TLM_MSG 0
```

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE telemetry messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 355 of file sample_mission_cfg.h.

12.29.2.10 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 386 of file sample_mission_cfg.h.

12.29.2.11 CFE_MISSION_ES_MEMSTATS_TLM_MSG

```
#define CFE_MISSION_ES_MEMSTATS_TLM_MSG 16
```

Definition at line 370 of file sample_mission_cfg.h.

12.29.2.12 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 403 of file sample_mission_cfg.h.

12.29.2.13 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 424 of file sample_mission_cfg.h.

12.29.2.14 CFE_MISSION_ES_SEND_HK_MSG

```
#define CFE_MISSION_ES_SEND_HK_MSG 8
```

Definition at line 318 of file sample_mission_cfg.h.

12.29.2.15 CFE_MISSION_EVS_CMD_MSG

```
#define CFE_MISSION_EVS_CMD_MSG 1
```

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE command messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 311 of file sample_mission_cfg.h.

12.29.2.16 CFE_MISSION_EVS_HK_TLM_MSG

```
#define CFE_MISSION_EVS_HK_TLM_MSG 1
```

Definition at line 356 of file sample_mission_cfg.h.

12.29.2.17 CFE_MISSION_EVS_LONG_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_LONG_EVENT_MSG_MSG 8
```

Definition at line 363 of file sample_mission_cfg.h.

12.29.2.18 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 259 of file sample_mission_cfg.h.

12.29.2.19 CFE_MISSION_EVS_SEND_HK_MSG

```
#define CFE_MISSION_EVS_SEND_HK_MSG 9
```

Definition at line 319 of file sample_mission_cfg.h.

12.29.2.20 CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG 9
```

Definition at line 364 of file sample_mission_cfg.h.

12.29.2.21 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 541 of file sample_mission_cfg.h.

12.29.2.22 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 515 of file sample_mission_cfg.h.

12.29.2.23 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 488 of file sample_mission_cfg.h.

12.29.2.24 CFE_MISSION_SB_ALLSUBS_TLM_MSG

```
#define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
```

Definition at line 368 of file sample_mission_cfg.h.

12.29.2.25 CFE_MISSION_SB_CMD_MSG

```
#define CFE_MISSION_SB_CMD_MSG 3
```

Definition at line 313 of file sample_mission_cfg.h.

12.29.2.26 CFE_MISSION_SB_HK_TLM_MSG

```
#define CFE_MISSION_SB_HK_TLM_MSG 3
```

Definition at line 358 of file sample_mission_cfg.h.

12.29.2.27 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 461 of file sample_mission_cfg.h.

12.29.2.28 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 53 of file sample_mission_cfg.h.

Referenced by HS_ValidateMATable().

12.29.2.29 CFE_MISSION_SB_ONESUB_TLM_MSG

```
#define CFE_MISSION_SB_ONESUB_TLM_MSG 14
```

Definition at line 369 of file sample_mission_cfg.h.

12.29.2.30 CFE_MISSION_SB_SEND_HK_MSG

```
#define CFE_MISSION_SB_SEND_HK_MSG 11
```

Definition at line 321 of file sample_mission_cfg.h.

12.29.2.31 CFE_MISSION_SB_STATS_TLM_MSG

```
#define CFE_MISSION_SB_STATS_TLM_MSG 10
```

Definition at line 365 of file sample_mission_cfg.h.

12.29.2.32 CFE_MISSION_SB_SUB_RPT_CTRL_MSG

```
#define CFE_MISSION_SB_SUB_RPT_CTRL_MSG 14
```

Definition at line 325 of file sample_mission_cfg.h.

12.29.2.33 CFE_MISSION_TBL_CMD_MSG

```
#define CFE_MISSION_TBL_CMD_MSG 4
```

Definition at line 314 of file sample_mission_cfg.h.

12.29.2.34 CFE_MISSION_TBL_HK_TLM_MSG

```
#define CFE_MISSION_TBL_HK_TLM_MSG 4
```

Definition at line 359 of file sample_mission_cfg.h.

12.29.2.35 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 444 of file sample_mission_cfg.h.

12.29.2.36 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 298 of file sample_mission_cfg.h.

12.29.2.37 CFE_MISSION_TBL_REG_TLM_MSG

```
#define CFE_MISSION_TBL_REG_TLM_MSG 12
```

Definition at line 367 of file sample_mission_cfg.h.

12.29.2.38 CFE_MISSION_TBL_SEND_HK_MSG

```
#define CFE_MISSION_TBL_SEND_HK_MSG 12
```

Definition at line 322 of file sample_mission_cfg.h.

12.29.2.39 CFE_MISSION_TEST_CMD_MSG

```
#define CFE_MISSION_TEST_CMD_MSG 2
```

Definition at line 312 of file sample_mission_cfg.h.

12.29.2.40 CFE_MISSION_TEST_HK_TLM_MSG

```
#define CFE_MISSION_TEST_HK_TLM_MSG 2
```

Definition at line 357 of file sample_mission_cfg.h.

12.29.2.41 CFE_MISSION_TIME_1HZ_CMD_MSG

```
#define CFE_MISSION_TIME_1HZ_CMD_MSG 17
```

Definition at line 328 of file sample_mission_cfg.h.

12.29.2.42 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 108 of file sample_mission_cfg.h.

12.29.2.43 CFE_MISSION_TIME_AT_TONE_WILL_BE

```
#define CFE_MISSION_TIME_AT_TONE_WILL_BE false
```

Definition at line 109 of file sample_mission_cfg.h.

12.29.2.44 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 72 of file sample_mission_cfg.h.

12.29.2.45 CFE_MISSION_TIME_CFG_DEFAULT_UTC

```
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
```

Definition at line 73 of file sample_mission_cfg.h.

12.29.2.46 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 85 of file sample_mission_cfg.h.

12.29.2.47 CFE_MISSION_TIME_CMD_MSG

```
#define CFE_MISSION_TIME_CMD_MSG 5
```

Definition at line 315 of file sample_mission_cfg.h.

12.29.2.48 CFE_MISSION_TIME_DATA_CMD_MSG

```
#define CFE_MISSION_TIME_DATA_CMD_MSG 0
```

Purpose cFE Portable Message Numbers for Global Messages

Description:

Portable message numbers for the cFE global messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 341 of file sample_mission_cfg.h.

12.29.2.49 CFE_MISSION_TIME_DEF_DELAY_SECS

```
#define CFE_MISSION_TIME_DEF_DELAY_SECS 0
```

Definition at line 167 of file sample_mission_cfg.h.

12.29.2.50 CFE_MISSION_TIME_DEF_DELAY_SUBS

```
#define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
```

Definition at line 168 of file sample_mission_cfg.h.

12.29.2.51 CFE_MISSION_TIME_DEF_LEAPS

```
#define CFE_MISSION_TIME_DEF_LEAPS 37
```

Definition at line 165 of file sample_mission_cfg.h.

12.29.2.52 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 159 of file sample_mission_cfg.h.

12.29.2.53 CFE_MISSION_TIME_DEF_MET_SUBS

```
#define CFE_MISSION_TIME_DEF_MET_SUBS 0
```

Definition at line 160 of file sample_mission_cfg.h.

12.29.2.54 CFE_MISSION_TIME_DEF_STCF_SECS

```
#define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
```

Definition at line 162 of file sample_mission_cfg.h.

12.29.2.55 CFE_MISSION_TIME_DEF_STCF_SUBS

```
#define CFE_MISSION_TIME_DEF_STCF_SUBS 0
```

Definition at line 163 of file sample_mission_cfg.h.

12.29.2.56 CFE_MISSION_TIME_DIAG_TLM_MSG

```
#define CFE_MISSION_TIME_DIAG_TLM_MSG 6
```

Definition at line 361 of file sample_mission_cfg.h.

12.29.2.57 CFE_MISSION_TIME_EPOCH_DAY

```
#define CFE_MISSION_TIME_EPOCH_DAY 1
```

Definition at line 186 of file sample_mission_cfg.h.

12.29.2.58 CFE_MISSION_TIME_EPOCH_HOUR

```
#define CFE_MISSION_TIME_EPOCH_HOUR 0
```

Definition at line 187 of file sample_mission_cfg.h.

12.29.2.59 CFE_MISSION_TIME_EPOCH_MICROS

```
#define CFE_MISSION_TIME_EPOCH_MICROS 0
```

Definition at line 190 of file sample_mission_cfg.h.

12.29.2.60 CFE_MISSION_TIME_EPOCH_MINUTE

```
#define CFE_MISSION_TIME_EPOCH_MINUTE 0
```

Definition at line 188 of file sample_mission_cfg.h.

12.29.2.61 CFE_MISSION_TIME_EPOCH_SECOND

```
#define CFE_MISSION_TIME_EPOCH_SECOND 0
```

Definition at line 189 of file sample_mission_cfg.h.

12.29.2.62 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 185 of file sample_mission_cfg.h.

12.29.2.63 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 228 of file sample_mission_cfg.h.

12.29.2.64 CFE_MISSION_TIME_HK_TLM_MSG

```
#define CFE_MISSION_TIME_HK_TLM_MSG 5
```

Definition at line 360 of file sample_mission_cfg.h.

12.29.2.65 CFE_MISSION_TIME_MAX_ELAPSED

```
#define CFE_MISSION_TIME_MAX_ELAPSED 200000
```

Definition at line 134 of file sample_mission_cfg.h.

12.29.2.66 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 133 of file sample_mission_cfg.h.

12.29.2.67 CFE_MISSION_TIME_SEND_CMD_MSG

```
#define CFE_MISSION_TIME_SEND_CMD_MSG 2
```

Definition at line 342 of file sample_mission_cfg.h.

12.29.2.68 CFE_MISSION_TIME_SEND_HK_MSG

```
#define CFE_MISSION_TIME_SEND_HK_MSG 13
```

Definition at line 323 of file sample_mission_cfg.h.

12.29.2.69 CFE_MISSION_TIME_TONE_CMD_MSG

```
#define CFE_MISSION_TIME_TONE_CMD_MSG 16
```

Definition at line 327 of file sample_mission_cfg.h.

12.30 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.

12.30.1 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:

12.30.2 Macro Definition Documentation

12.30.2.1 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.

12.30.2.2 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.

12.30.2.3 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.

12.30.2.4 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.

12.30.2.5 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.

12.30.2.6 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.

12.30.2.7 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.

12.30.2.8 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.

12.30.2.9 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.

12.30.2.10 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.

12.30.2.11 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.

12.30.2.12 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.

12.30.2.13 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.

12.31 cfe/docs/src/cfe_api.dox File Reference

12.32 cfe/docs/src/cfe_es.dox File Reference

12.33 cfe/docs/src/cfe_evs.dox File Reference

12.34 cfe/docs/src/cfe_frontpage.dox File Reference

12.35 cfe/docs/src/cfe_glossary.dox File Reference

12.36 cfe/docs/src/cfe_sb.dox File Reference

12.37 cfe/docs/src/cfe_tbl.dox File Reference

12.38 cfe/docs/src/cfe_time.dox File Reference

12.39 cfe/docs/src/cfe_xref.dox File Reference

12.40 cfe/docs/src/cfs_versions.dox File Reference

12.41 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"
```

12.41.1 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.

12.42 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.
- `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.

12.42.1 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

12.42.2 Function Documentation

12.42.2.1 CFE_Config_getIdByName()

```
CFE_ConfigId_t CFE_Config_getIdByName (
    const char * Name )
```

Obtain the ID value associated with a configuration name.

Parameters

in	Name	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
-------------------------------	---

12.42.2.2 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

12.42.2.3 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

NULL	if key is not defined or not set
------	----------------------------------

12.42.2.4 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

12.42.2.5 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
---	----------------------------------

12.42.2.6 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

12.43 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_api_typedefs.h"
```

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)

12.43.1 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

12.43.2 Macro Definition Documentation

12.43.2.1 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.

12.43.2.2 CFE_CONFIGID_UNDEFINED

```
#define CFE_CONFIGID_UNDEFINED CFE_CONFIGID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 49 of file cfe_config_api_typedefs.h.

12.43.3 Typedef Documentation

12.43.3.1 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.

12.43.3.2 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.

12.44 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference

```
#include "common_types.h"
```

Macros

- `#define CFE_MAKE_BIG16(n) (((n) << 8) & 0xFF00) | (((n) >> 8) & 0x00FF)`
- `#define CFE_MAKE_BIG32(n) (((n) << 24) & 0xFF000000) | (((n) << 8) & 0x00FF0000) | (((n) >> 8) & 0x0000FF00) | (((n) >> 24) & 0x000000FF)`

12.44.1 Detailed Description

Purpose: Define macros to enforce big-endian/network byte order for 16 and 32 bit integers

12.44.2 Macro Definition Documentation

12.44.2.1 CFE_MAKE_BIG16

```
#define CFE_MAKE_BIG16(  
    n ) (((n) << 8) & 0xFF00) | (((n) >> 8) & 0x00FF))
```

Definition at line 64 of file cfe_endian.h.

12.44.2.2 CFE_MAKE_BIG32

```
#define CFE_MAKE_BIG32(  
    n ) (((n) << 24) & 0xFF000000) | (((n) << 8) & 0x00FF0000) | (((n) >> 8) & 0x0000FF00) | (((n) >> 24) & 0x000000FF))
```

Definition at line 65 of file cfe_endian.h.

12.45 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference

```
#include "osapi.h"
```

Macros

- #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_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)

- `#define CFE_ES_APP_FILTER_OVERLOAD ((CFE_Status_t)0xc2000004)`
Application Filter Overload.
- `#define CFE_ES_RESET_AREA_POINTER ((CFE_Status_t)0xc2000005)`
Reset Area Pointer Failure.
- `#define CFE_ES_EVT_NOT_REGISTERED ((CFE_Status_t)0xc2000006)`
Event Not Registered.
- `#define CFE_ES_FILE_WRITE_ERROR ((CFE_Status_t)0xc2000007)`
File Write Error.
- `#define CFE_ES_INVALID_PARAMETER ((CFE_Status_t)0xc2000008)`
Invalid Pointer.
- `#define CFE_ES_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)
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)

- `#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)

- `#define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)`
Partial Load Warning.
- `#define CFE_TBL_INFO_DUMP_PENDING ((CFE_Status_t)0x4c000024)`
Partial Load Error.
- `#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`

12.45.1 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

12.45.2 Macro Definition Documentation

12.45.2.1 CFE_EVENTS_SERVICE

```
#define CFE_EVENTS_SERVICE ((CFE_Status_t)0x02000000)
```

Event Service.

Definition at line 97 of file cfe_error.h.

12.45.2.2 CFE_EXECUTIVE_SERVICE

```
#define CFE_EXECUTIVE_SERVICE ((CFE_Status_t)0x04000000)
```

Executive Service.

Definition at line 98 of file cfe_error.h.

12.45.2.3 CFE_FILE_SERVICE

```
#define CFE_FILE_SERVICE ((CFE_Status_t)0x06000000)
```

File Service.

Definition at line 99 of file cfe_error.h.

12.45.2.4 CFE_GENERIC_SERVICE

```
#define CFE_GENERIC_SERVICE ((CFE_Status_t)0x08000000)
```

Generic Service.

Definition at line 100 of file cfe_error.h.

12.45.2.5 CFE_SERVICE_BITMASK

```
#define CFE_SERVICE_BITMASK ((CFE_Status_t)0x0e000000)
```

Error Service Bitmask.

Definition at line 95 of file cfe_error.h.

12.45.2.6 CFE_SEVERITY_BITMASK

```
#define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)
```

Error Severity Bitmask.

Definition at line 86 of file cfe_error.h.

12.45.2.7 CFE_SEVERITY_ERROR

```
#define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)
```

Severity Error.

Definition at line 90 of file cfe_error.h.

12.45.2.8 CFE_SEVERITY_INFO

```
#define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)
```

Severity Info.

Definition at line 89 of file cfe_error.h.

12.45.2.9 CFE_SEVERITY_SUCCESS

```
#define CFE_SEVERITY_SUCCESS ((CFE_Status_t)0x00000000)
```

Severity Success.

Definition at line 88 of file cfe_error.h.

12.45.2.10 CFE_SOFTWARE_BUS_SERVICE

```
#define CFE_SOFTWARE_BUS_SERVICE ((CFE_Status_t)0x0a000000)
```

Software Bus Service.

Definition at line 101 of file cfe_error.h.

12.45.2.11 CFE_TABLE_SERVICE

```
#define CFE_TABLE_SERVICE ((CFE_Status_t)0x0c000000)
```

Table Service.

Definition at line 102 of file cfe_error.h.

12.45.2.12 CFE_TIME_SERVICE

```
#define CFE_TIME_SERVICE ((CFE_Status_t)0x0e000000)
```

Time Service.

Definition at line 103 of file cfe_error.h.

12.45.3 Typedef Documentation

12.45.3.1 CFE_Status_t

```
typedef int32 CFE_Status_t
```

Definition at line 43 of file cfe_error.h.

12.46 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)`
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)`

- `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.
- `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.
- `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, uint32 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.*

12.46.1 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:

12.46.2 Macro Definition Documentation

12.46.2.1 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`.

12.46.2.2 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.

12.46.2.3 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.

12.46.2.4 OS_PRINTF

```
#define OS_PRINTF(
    m,
    n )
```

Definition at line 50 of file cfe_es.h.

12.47 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_CDSHANDLE_C(val) ((CFE_ES_CDSHandle_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 union CFE_ES_PoolAlign CFE_ES_PoolAlign_t`
Pool Alignment.
- `typedef void * CFE_ES_MemPoolBuf_t`
Pointer type used for memory pool API.

12.47.1 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:

12.47.2 Macro Definition Documentation

12.47.2.1 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.

12.47.2.2 CFE_ES_APPID_C

```
#define CFE_ES_APPID_C( val ) ((CFE_ES_AppId_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 156 of file cfe_es_api_typedefs.h.

12.47.2.3 CFE_ES_APPID_UNDEFINED

```
#define CFE_ES_APPID_UNDEFINED CFE_ES_APPID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 168 of file cfe_es_api_typedefs.h.

Referenced by HS_HousekeepingReq(), HS_MonitorApplications(), and HS_MonitorEvent().

12.47.2.4 CFE_ES_CDS_BAD_HANDLE

```
#define CFE_ES_CDS_BAD_HANDLE CFE_ES_CDSHANDLE_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 173 of file cfe_es_api_typedefs.h.

12.47.2.5 CFE_ES_CDSHANDLE_C

```
#define CFE_ES_CDSHANDLE_C(  
    val ) ((CFE_ES_CDSHandle_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 161 of file cfe_es_api_typedefs.h.

12.47.2.6 CFE_ES_COUNTERID_C

```
#define CFE_ES_COUNTERID_C(  
    val ) ((CFE_ES_CounterId_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 159 of file cfe_es_api_typedefs.h.

12.47.2.7 CFE_ES_COUNTERID_UNDEFINED

```
#define CFE_ES_COUNTERID_UNDEFINED CFE_ES_COUNTERID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 171 of file cfe_es_api_typedefs.h.

Referenced by HS_HousekeepingReq().

12.47.2.8 CFE_ES_LIBID_C

```
#define CFE_ES_LIBID_C(  
    val ) ((CFE_ES_LibId_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 158 of file cfe_es_api_typedefs.h.

12.47.2.9 CFE_ES_LIBID_UNDEFINED

```
#define CFE_ES_LIBID_UNDEFINED CFE_ES_LIBID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 170 of file cfe_es_api_typedefs.h.

12.47.2.10 CFE_ES_MEMHANDLE_C

```
#define CFE_ES_MEMHANDLE_C(  
    val ) ((CFE_ES_MemHandle_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 160 of file cfe_es_api_typedefs.h.

12.47.2.11 CFE_ES_MEMHANDLE_UNDEFINED

```
#define CFE_ES_MEMHANDLE_UNDEFINED CFE_ES_MEMHANDLE_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 172 of file cfe_es_api_typedefs.h.

12.47.2.12 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 141 of file cfe_es_api_typedefs.h.

12.47.2.13 CFE_ES_NO_MUTEX

```
#define CFE_ES_NO_MUTEX false
```

Indicates that the memory pool selection will not use a semaphore.

Definition at line 188 of file cfe_es_api_typedefs.h.

12.47.2.14 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 108 of file cfe_es_api_typedefs.h.

12.47.2.15 CFE_ES_TASK_STACK_ALLOCATE

```
#define CFE_ES_TASK_STACK_ALLOCATE NULL /* aka OS_TASK_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 185 of file cfe_es_api_typedefs.h.

12.47.2.16 CFE_ES_TASKID_C

```
#define CFE_ES_TASKID_C(  
    val ) ((CFE_ES_TaskId_t)CFE_RESOURCEID_WRAP(val))
```

Definition at line 157 of file cfe_es_api_typedefs.h.

12.47.2.17 CFE_ES_TASKID_UNDEFINED

```
#define CFE_ES_TASKID_UNDEFINED CFE_ES_TASKID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 169 of file cfe_es_api_typedefs.h.

Referenced by HS_HousekeepingReq().

12.47.2.18 CFE_ES_USE_MUTEX

```
#define CFE_ES_USE_MUTEX true
```

Indicates that the memory pool selection will use a semaphore.

Definition at line 189 of file cfe_es_api_typedefs.h.

12.47.3 Typedef Documentation

12.47.3.1 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.

12.47.3.2 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.

12.47.3.3 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 129 of file cfe_es_api_typedefs.h.

12.47.3.4 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.

12.47.3.5 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.

12.47.3.6 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.

12.48 cfe/modules/core_api/fsw/inc/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))
- #define **CFE_ES_MEMADDRESS_C**(x) ((CFE_ES_MemAddress_t)((cpuaddr)(x)&0xFFFFFFFF))

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.

12.48.1 Detailed Description

Declarations and prototypes for `cfe_es_extern_typedefs` module

12.48.2 Macro Definition Documentation

12.48.2.1 `CFE_ES_MEMADDRESS_C`

```
#define CFE_ES_MEMADDRESS_C(  
    x ) ((CFE_ES_MemAddress_t)((cpuaddr) (x)&0xFFFFFFFF))
```

Definition at line 426 of file `cfe_es_extern_typedefs.h`.

12.48.2.2 CFE_ES_MEMOFFSET_C

```
#define CFE_ES_MEMOFFSET_C( x ) ( (CFE_ES_MemOffset_t)(x) )
```

Definition at line 395 of file cfe_es_extern_typedefs.h.

12.48.3 TYPEDEF Documentation

12.48.3.1 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 329 of file cfe_es_extern_typedefs.h.

12.48.3.2 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.

12.48.3.3 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 322 of file cfe_es_extern_typedefs.h.

12.48.3.4 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 117 of file `cfe_es_extern_typedefs.h`.

12.48.3.5 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.

12.48.3.6 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 365 of file `cfe_es_extern_typedefs.h`.

12.48.3.7 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.

12.48.3.8 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 350 of file cfe_es_extern_typedefs.h.

12.48.3.9 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 88 of file cfe_es_extern_typedefs.h.

12.48.3.10 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 343 of file cfe_es_extern_typedefs.h.

12.48.3.11 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 268 of file cfe_es_extern_typedefs.h.

12.48.3.12 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 64 of file `cfe_es_extern_typedefs.h`.

12.48.3.13 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 416 of file `cfe_es_extern_typedefs.h`.

12.48.3.14 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 358 of file `cfe_es_extern_typedefs.h`.

12.48.3.15 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 389 of file cfe_es_extern_typedefs.h.

12.48.3.16 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](#)

12.48.3.17 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 186 of file cfe_es_extern_typedefs.h.

12.48.3.18 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 244 of file `cfe_es_extern_typedefs.h`.

12.48.3.19 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 336 of file `cfe_es_extern_typedefs.h`.

12.48.3.20 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.

12.48.3.21 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 375 of file cfe_es_extern_typedefs.h.

12.48.4 Enumeration Type Documentation

12.48.4.1 CFE_ES_AppState

```
enum CFE_ES_AppState
```

Label definitions associated with CFE_ES_AppState_Enum_t.

Enumerator

CFE_ES_AppState_UNDEFINED	Initial state before app thread is started.
CFE_ES_AppState_EARLY_INIT	App thread has started, app performing early initialization of its own data.
CFE_ES_AppState_LATE_INIT	Early/Local initialization is complete. First sync point.
CFE_ES_AppState_RUNNING	All initialization is complete. Second sync point.
CFE_ES_AppState_WAITING	Application is waiting on a Restart/Reload/Delete request.
CFE_ES_AppState_STOPPED	Application is stopped.
CFE_ES_AppState_MAX	Reserved entry, marker for the maximum state.

Definition at line 273 of file cfe_es_extern_typedefs.h.

12.48.4.2 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 93 of file cfe_es_extern_typedefs.h.

12.48.4.3 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 69 of file cfe_es_extern_typedefs.h.

12.48.4.4 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 249 of file cfe_es_extern_typedefs.h.

12.48.4.5 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 45 of file cfe_es_extern_typedefs.h.

12.48.4.6 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.
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 122 of file cfe_es_extern_typedefs.h.

12.48.4.7 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 191 of file cfe_es_extern_typedefs.h.

12.49 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, uint16 EventType, const char *Spec,...) OS_PRINTF(3

Generate a software event.
- CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (uint16 EventID, uint16 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, uint16 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.

12.49.1 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

12.49.2 Macro Definition Documentation

12.49.2.1 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.

12.49.2.2 CFE_EVS_SendCrit

```
#define CFE_EVS_SendCrit(  
    E,  
    ... ) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)
```

Definition at line 50 of file cfe_evs.h.

12.49.2.3 CFE_EVS_SendDbg

```
#define CFE_EVS_SendDbg(  
    E,  
    ... ) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)
```

Definition at line 47 of file cfe_evs.h.

12.49.2.4 CFE_EVS_SendErr

```
#define CFE_EVS_SendErr(  
    E,  
    ... ) CFE_EVS_Send(E, ERROR, __VA_ARGS__)
```

Definition at line 49 of file cfe_evs.h.

12.49.2.5 CFE_EVS_SendInfo

```
#define CFE_EVS_SendInfo(  
    E,  
    ... ) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)
```

Definition at line 48 of file cfe_evs.h.

12.50 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](#) 0xFFFF8
Sends the first 8 events. All remaining messages are filtered.
- #define [CFE_EVS_FIRST_16_STOP](#) 0xFFFF0
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.

12.50.1 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

12.50.2 Macro Definition Documentation

12.50.2.1 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.

12.50.2.2 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.

12.50.2.3 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.

12.50.2.4 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.

12.50.2.5 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.

12.50.2.6 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.

12.50.2.7 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.

12.50.2.8 CFE_EVS_FIRST_8_STOP

```
#define CFE_EVS_FIRST_8_STOP 0xFFFF8
```

Sends the first 8 events. All remaining messages are filtered.

Definition at line 48 of file cfe_evs_api_typedefs.h.

12.50.2.9 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.

12.50.2.10 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.

12.50.2.11 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.

12.50.3 Typedef Documentation

12.50.3.1 CFE_EVS_BinFilter_t

```
typedef struct CFE_EVS_BinFilter CFE_EVS_BinFilter_t
```

Event message filter definition structure.

12.51 cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Typedefs

- **typedef uint8 CFE_EVS_MsgFormat_Enum_t**
Identifies format of log messages.
- **typedef uint8 CFE_EVS_LogMode_Enum_t**
Identifies handling of log messages after storage is filled.
- **typedef uint16 CFE_EVS_EventType_Enum_t**
Identifies type of event message.
- **typedef uint8 CFE_EVS_EventFilter_Enum_t**
Identifies event filter schemes.
- **typedef uint8 CFE_EVS_EventOutput_Enum_t**
Identifies event output port.

Enumerations

- enum **CFE_EVS_MsgFormat** { **CFE_EVS_MsgFormat_SHORT** = 0, **CFE_EVS_MsgFormat_LONG** = 1 }
Label definitions associated with CFE_EVS_MsgFormat_Enum_t.
- enum **CFE_EVS_LogMode** { **CFE_EVS_LogMode_OVERWRITE** = 0, **CFE_EVS_LogMode_DISCARD** = 1 }
Label definitions associated with CFE_EVS_LogMode_Enum_t.
- enum **CFE_EVS_EventType** { **CFE_EVS_EventType_DEBUG** = 1, **CFE_EVS_EventType_INFORMATION** = 2, **CFE_EVS_EventType_ERROR** = 3, **CFE_EVS_EventType_CRITICAL** = 4 }
Label definitions associated with CFE_EVS_EventType_Enum_t.
- enum **CFE_EVS_EventFilter** { **CFE_EVS_EventFilter_BINARY** = 0 }
Label definitions associated with CFE_EVS_EventFilter_Enum_t.
- enum **CFE_EVS_EventOutput** { **CFE_EVS_EventOutput_PORT1** = 1, **CFE_EVS_EventOutput_PORT2** = 2, **CFE_EVS_EventOutput_PORT3** = 3, **CFE_EVS_EventOutput_PORT4** = 4 }
Label definitions associated with CFE_EVS_EventOutput_Enum_t.

12.51.1 Detailed Description

Declarations and prototypes for cfe_evs_extern_typedefs module

12.51.2 Typedef Documentation

12.51.2.1 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 139 of file cfe_evs_extern_typedefs.h.

12.51.2.2 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 173 of file cfe_evs_extern_typedefs.h.

12.51.2.3 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 120 of file cfe_evs_extern_typedefs.h.

12.51.2.4 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 86 of file cfe_evs_extern_typedefs.h.

12.51.2.5 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 62 of file cfe_evs_extern_typedefs.h.

12.51.3 Enumeration Type Documentation

12.51.3.1 CFE_EVS_EventFilter

enum [CFE_EVS_EventFilter](#)

Label definitions associated with CFE_EVS_EventFilter_Enum_t.

Enumerator

CFE_EVS_EventFilter_BINARY	Binary event filter.
----------------------------	----------------------

Definition at line 125 of file cfe_evs_extern_typedefs.h.

12.51.3.2 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 144 of file cfe_evs_extern_typedefs.h.

12.51.3.3 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 91 of file cfe_evs_extern_typedefs.h.

12.51.3.4 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 67 of file cfe_evs_extern_typedefs.h.

12.51.3.5 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 43 of file cfe_evs_extern_typedefs.h.

12.52 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_BackgroundFileDumplsPending \(const CFE_FS_FileWriteMetaData_t *Meta\)](#)
Query if a background file write request is currently pending.

12.52.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.53 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference

```
#include "common_types.h"
#include "osconfig.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_UNDEFINED](#), [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](#) }

12.53.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.53.2 Typedef Documentation

12.53.2.1 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

<i>in, out</i>	<i>Meta</i>	Pointer to the metadata object
<i>in</i>	<i>RecordNum</i>	Incrementing record number counter
<i>out</i>	<i>Buffer</i>	Pointer to buffer data block, should be set by implementation
<i>out</i>	<i>BufSize</i>	Pointer to buffer data size, should be set by implementation

Returns

End of file status

Return values

<i>true</i>	if at last data record, and output file should be closed
<i>false</i>	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.

12.53.2.2 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).

12.53.2.3 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.

12.53.3 Enumeration Type Documentation

12.53.3.1 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

CFE_FS_FileCategory_UNKNOWN	Placeholder, unknown file category
CFE_FS_FileCategory_DYNAMIC_MODULE	Dynamically loadable apps/libraries (e.g. .so, .o, .dll, etc)
CFE_FS_FileCategory_BINARY_DATA_DUMP	Binary log file generated by various data dump commands
CFE_FS_FileCategory_TEXT_LOG	Text-based log file generated by various commands
CFE_FS_FileCategory_SCRIPT	Text-based Script files (e.g. ES startup script)
CFE_FS_FileCategory_TEMP	Temporary/Ephemeral files
CFE_FS_FileCategory_MAX	Placeholder, keep last

Definition at line 48 of file cfe_fs_api_typedefs.h.

12.53.3.2 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 68 of file cfe_fs_api_typedefs.h.

12.54 cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h File Reference

#include "common_types.h"

Data Structures

- struct [CFE_FS_Header](#)
Standard cFE File header structure definition.

Macros

- #define [CFE_FS_HDR_DESC_MAX_LEN](#) 32
Max length of description field in a standard cFE File Header.
- #define [CFE_FS_FILE_CONTENT_ID](#) 0x63464531
Magic Number for cFE compliant files (= 'cFE1')

Typedefs

- typedef uint32 [CFE_FS_SubType_Enum_t](#)
Content descriptor for File Headers.
- 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.

12.54.1 Detailed Description

Declarations and prototypes for cfe_fs_extern_typedefs module

12.54.2 Macro Definition Documentation

12.54.2.1 CFE_FS_FILE_CONTENT_ID

```
#define CFE_FS_FILE_CONTENT_ID 0x63464531
```

Magic Number for cFE compliant files (= 'cFE1')

Definition at line 51 of file cfe_fs_extern_typedefs.h.

12.54.2.2 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 49 of file cfe_fs_extern_typedefs.h.

12.54.3 Typedef Documentation

12.54.3.1 CFE_FS_Header_t

```
typedef struct CFE_FS_Header CFE_FS_Header_t
```

Standard cFE File header structure definition.

12.54.3.2 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 199 of file cfe_fs_extern_typedefs.h.

12.54.4 Enumeration Type Documentation

12.54.4.1 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

CFE_FS_SubType_ES_ERLOG	Executive Services Exception/Reset Log Type. Executive Services Exception/Reset Log File which is generated in response to a \$sc_\$cpu_ES_WriteERLog2File command.
CFE_FS_SubType_ES_SYSLOG	Executive Services System Log Type. Executive Services System Log File which is generated in response to a \$sc_\$cpu_ES_WriteSysLog2File command.
CFE_FS_SubType_ES_QUERYALL	Executive Services Information on All Applications File. Executive Services Information on All Applications File which is generated in response to a \$sc_\$cpu_ES_WriteAppInfo2File command.
CFE_FS_SubType_ES_PERFDATA	Executive Services Performance Data File. Executive Services Performance Analyzer Data File which is generated in response to a \$sc_\$cpu_ES_StopLAData command.
CFE_FS_SubType_ES_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.

Enumerator

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 61 of file `cfe_fs_extern_typedefs.h`.

12.55 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_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)
- `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.

- 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)
 - 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.

12.55.1 Detailed Description

Message access APIs

12.56 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**
Message sequence count.
- **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 union 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.

12.56.1 Detailed Description

Typedefs for Message API

- Separate from API so these can be adjusted for custom implementations

12.56.2 Macro Definition Documentation

12.56.2.1 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.

12.56.2.2 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.

12.56.2.3 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.

12.56.3 Typedef Documentation

12.56.3.1 CFE_MSG_ApId_t

```
typedef uint16 CFE_MSG_ApId_t
```

Message application ID.

Definition at line 50 of file cfe_msg_api_typedefs.h.

12.56.3.2 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.

12.56.3.3 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.

12.56.3.4 CFE_MSG_EDSVersion_t

```
typedef uint16 CFE_MSG_EDSVersion_t
```

Message EDS version.

Definition at line 52 of file cfe_msg_api_typedefs.h.

12.56.3.5 CFE_MSG_Endian_t

```
typedef enum CFE_MSG_Endian CFE_MSG_Endian_t
```

Endian flag.

12.56.3.6 CFE_MSG_FcnCode_t

```
typedef uint16 CFE_MSG_FcnCode_t
```

Message function code.

Definition at line 48 of file cfe_msg_api_typedefs.h.

12.56.3.7 CFE_MSG_HeaderVersion_t

```
typedef uint16 CFE_MSG_HeaderVersion_t
```

Message header version.

Definition at line 49 of file cfe_msg_api_typedefs.h.

12.56.3.8 CFE_MSG_Message_t

```
typedef union CFE_MSG_Message CFE_MSG_Message_t
```

cFS generic base message

Definition at line 102 of file cfe_msg_api_typedefs.h.

12.56.3.9 CFE_MSG_PlaybackFlag_t

```
typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t
```

Playback flag.

12.56.3.10 CFE_MSG_SegmentationFlag_t

```
typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t
```

Segmentation flags.

12.56.3.11 CFE_MSG_SequenceCount_t

```
typedef uint16 CFE_MSG_SequenceCount_t
```

Message sequence count.

Definition at line 51 of file cfe_msg_api_typedefs.h.

12.56.3.12 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.

12.56.3.13 CFE_MSG_Subsystem_t

```
typedef uint16 CFE_MSG_Subsystem_t
```

Message subsystem.

Definition at line 53 of file cfe_msg_api_typedefs.h.

12.56.3.14 CFE_MSG_System_t

```
typedef uint16 CFE_MSG_System_t
```

Message system.

Definition at line 54 of file cfe_msg_api_typedefs.h.

12.56.3.15 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.

12.56.3.16 CFE_MSG_Type_t

```
typedef enum CFE_MSG_Type CFE_MSG_Type_t
```

Message type.

12.56.4 Enumeration Type Documentation

12.56.4.1 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`.

12.56.4.2 CFE_MSG_PlaybackFlag

```
enum CFE\_MSG\_PlaybackFlag
```

Playback flag.

Enumerator

<code>CFE_MSG_PlayFlag_Invalid</code>	Invalid playback setting.
<code>CFE_MSG_PlayFlag_Original</code>	Original.
<code>CFE_MSG_PlayFlag_Playback</code>	Playback.

Definition at line 83 of file `cfe_msg_api_typedefs.h`.

12.56.4.3 CFE_MSG_SegmentationFlag

```
enum CFE\_MSG\_SegmentationFlag
```

Segmentation flags.

Enumerator

<code>CFE_MSG_SegFlag_Invalid</code>	Invalid segmentation flag.
<code>CFE_MSG_SegFlag_Continue</code>	Continuation segment of User Data.
<code>CFE_MSG_SegFlag_First</code>	First segment of User Data.
<code>CFE_MSG_SegFlag_Last</code>	Last segment of User Data.
<code>CFE_MSG_SegFlag_Unsegmented</code>	Unsegmented data.

Definition at line 65 of file `cfe_msg_api_typedefs.h`.

12.56.4.4 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.

12.57 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_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.

Resource ID test/conversion macros and inline functions

- `#define CFE_RESOURCEID_TO ULONG(id) CFE_ResourceId_ToInteger(CFE_RESOURCEID_UNWRAP(id))`
Convert a derived (app-specific) ID directly into an "unsigned long".
- `#define CFE_RESOURCEID_TEST_DEFINED(id) CFE_ResourceId_IsDefined(CFE_RESOURCEID_UNWRAP(id))`
Determine if a derived (app-specific) ID is defined or not.
- `#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.
- `static unsigned long CFE_ResourceId_ToInteger (CFE_ResourceId_t id)`
Convert a resource ID to an integer.
- `static CFE_ResourceId_t CFE_ResourceId_FromInteger (unsigned long Value)`
Convert an integer to a resource ID.
- `static bool CFE_ResourceId_Equal (CFE_ResourceId_t id1, CFE_ResourceId_t id2)`
Compare two Resource ID values for equality.
- `static bool CFE_ResourceId_IsDefined (CFE_ResourceId_t id)`
Check if a resource ID value is defined.

12.57.1 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)

12.57.2 Macro Definition Documentation

12.57.2.1 CFE_RESOURCEID_TEST_DEFINED

```
#define CFE_RESOURCEID_TEST_DEFINED(  
    id ) CFE_ResourceId_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.

12.57.2.2 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.

12.57.2.3 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.

12.57.3 Function Documentation

12.57.3.1 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\(\)](#).

12.57.3.2 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.
--	------------------------------------

Referenced by [CFE_Resourceld_IsDefined\(\)](#).

12.57.3.3 CFE_Resourceld_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_Resourceld_ToInteger\(\)](#), and reconstitutes the original CFE_Resourceld_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_Resourceld_ToInteger\(\)](#)

Parameters

in	Value	Integer value to convert
----	-------	--------------------------

Returns

ID value corresponding to integer

Definition at line 121 of file cfe_resourceid.h.

12.57.3.4 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	Resource Id	the resource ID to decode
----	----------------	---------------------------

Returns

The base value associated with that ID

Referenced by CFE_ResourceId_IsDefined().

12.57.3.5 CFE_ResourceId_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>Resource← Id</i>	the resource ID to decode
----	-------------------------	---------------------------

Returns

The serial number associated with that ID

Referenced by CFE_Resourceld_IsDefined().

12.57.3.6 CFE_Resourceld_IsDefined()

```
static bool CFE_Resourceld_IsDefined (
    CFE_Resourceld_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_Resourceld_IsDefined(CFE_RESOURCEID_UNDEFINED)
```

Always returns false.

Parameters

in	<i>id</i>	Resource ID to check
----	-----------	----------------------

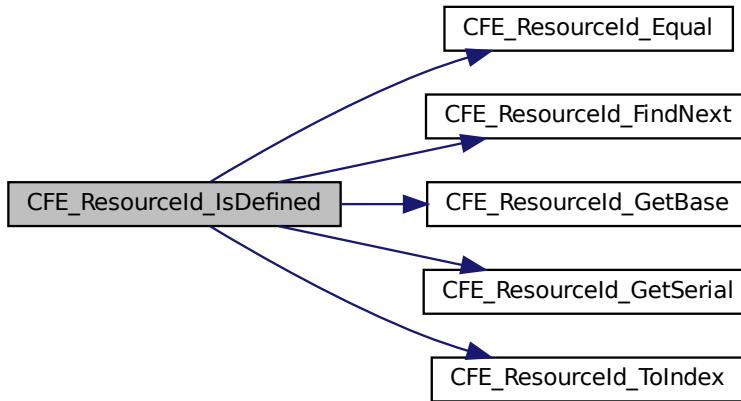
Returns

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_Resourceld_Equal(), CFE_Resourceld_FindNext(), CFE_Resourceld_GetBase(), CFE_Resource←
Id_GetSerial(), CFE_Resourceld_ToIndex(), and CFE_RESOURCEID_UNDEFINED.

Here is the call graph for this function:



12.57.3.7 CFE_ResourceId_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.

Referenced by CFE_Resourceld_IsDefined().

12.57.3.8 CFE_Resourceld_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_Resourceld_ToIndex\(\)](#) for cases where a zero-based array/table index is needed.

See also

[CFE_Resourceld_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.

12.58 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.

12.58.1 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)

12.58.2 Macro Definition Documentation

12.58.2.1 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.

12.58.2.2 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_Resourceld_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_Resourceld_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.

Referenced by CFE_Resourceld_IsDefined().

12.59 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 Optis)`
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 IncrementSequenceCount)`
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 IncrementSequenceCount)`
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`.

12.59.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.

Author: R.McGraw/SSI

12.59.2 Macro Definition Documentation

12.59.2.1 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.

12.59.2.2 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.

Referenced by HS_MonitorApplications().

12.59.2.3 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.

Referenced by HS_AppMonStatusRefresh().

12.59.2.4 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.

12.60 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.

12.60.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.

Author: R.McGraw/SSI

12.60.2 Macro Definition Documentation

12.60.2.1 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`.

Referenced by `HS_AppMain()`, and `HS_EnableEventMonCmd()`.

12.60.2.2 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`.

Referenced by `HS_AppPipe()`, `HS_ValidateMATable()`, and `HS_VerifyMsgLength()`.

12.60.2.3 CFE_SB_INVALID_PIPE

```
#define CFE_SB_INVALID_PIPE CFE_SB_PIPEID_C(CFE_RESOURCEID_UNDEFINED)
```

A `CFE_SB_Pipeld_t` value which is always invalid.

This may be used as a safe initializer for `CFE_SB_Pipeld_t` values

Definition at line 125 of file `cfe_sb_api_typedefs.h`.

12.60.2.4 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`.

Referenced by [CFE_SB_ValueToMsgId\(\)](#).

12.60.2.5 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`.

12.60.2.6 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`.

Referenced by [CFE_SB_MsgId_Equal\(\)](#), and [CFE_SB_MsgIdToValue\(\)](#).

12.60.2.7 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`.

12.60.2.8 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`.

12.60.2.9 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`.

12.60.2.10 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](#).

Referenced by [HS_ProcessCommands\(\)](#).

12.60.2.11 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](#).

12.60.2.12 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](#).

12.60.3 Typedef Documentation

12.60.3.1 CFE_SB_Buffer_t

```
typedef union CFE_SB_Msg CFE_SB_Buffer_t
```

Software Bus generic message.

12.61 cfe/modules/core_api/fsw/inc/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_Pipeld_t](#)
CFE_SB_Pipeld_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.

12.61.1 Detailed Description

Declarations and prototypes for cfe_sb_extern_typeDefs module

12.61.2 Macro Definition Documentation

12.61.2.1 CFE_SB_SUB_ENTRIES_PER_PKT

```
#define CFE_SB_SUB_ENTRIES_PER_PKT 20
```

Configuration parameter used by SBN App.

Definition at line 42 of file cfe_sb_extern_typedefs.h.

12.61.3 Typedef Documentation

12.61.3.1 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_S←B_HIGHEST_VALID_MSGID](#)

Definition at line 103 of file cfe_sb_extern_typedefs.h.

12.61.3.2 CFE_SB_Pipeld_t

```
typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t
```

CFE_SB_Pipeld_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 126 of file cfe_sb_extern_typedefs.h.

12.61.3.3 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 66 of file cfe_sb_extern_typedefs.h.

12.61.3.4 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 90 of file `cfe_sb_extern_typedefs.h`.

12.61.3.5 CFE_SB_Routeld_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 95 of file `cfe_sb_extern_typedefs.h`.

12.61.4 Enumeration Type Documentation

12.61.4.1 CFE_SB_QosPriority

```
enum CFE_SB_QosPriority
```

Label definitions associated with `CFE_SB_QosPriority_Enum_t`.

Enumerator

CFE_SB_QosPriority_LOW	Normal priority level.
CFE_SB_QosPriority_HIGH	High priority.

Definition at line 47 of file `cfe_sb_extern_typedefs.h`.

12.61.4.2 CFE_SB_QosReliability

```
enum CFE_SB_QosReliability
```

Label definitions associated with `CFE_SB_QosReliability_Enum_t`.

Enumerator

CFE_SB_QosReliability_LOW	Normal (best-effort) reliability.
CFE_SB_QosReliability_HIGH	High reliability.

Definition at line 71 of file cfe_sb_extern_typedefs.h.

12.62 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.

12.62.1 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:

12.63 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)

- `#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.

12.63.1 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:

12.63.2 Macro Definition Documentation

12.63.2.1 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.

12.63.2.2 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.

12.63.3 Typedef Documentation

12.63.3.1 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.

12.63.3.2 CFE_TBL_Handle_t

```
typedef int16 CFE_TBL_Handle_t
```

Table Handle primitive.

Definition at line 87 of file cfe_tbl_api_typedefs.h.

12.63.3.3 CFE_TBL_Info_t

```
typedef struct CFE_TBL_Info CFE_TBL_Info_t
```

Table Info.

12.63.3.4 CFE_TBL_SrcEnum_t

```
typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t
```

Table Source.

12.63.4 Enumeration Type Documentation

12.63.4.1 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 <code>Size</code> parameter.

Definition at line 90 of file `cfe_tbl_api_typedefs.h`.

12.64 cfe/modules/core_api/fsw/inc/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.

12.64.1 Detailed Description

Declarations and prototypes for cfe_tbl_extern_TypeDef module

12.64.2 TYPEDef Documentation

12.64.2.1 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 64 of file cfe_tbl_extern_TypeDef.h.

12.64.2.2 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.

12.64.3 Enumeration Type Documentation

12.64.3.1 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 45 of file cfe_tbl_extern_typedefs.h.

12.65 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](#)

Macros

- #define [CFE_TBL_FILEDEF](#)(ObjName, TblName, Desc, Filename)

Typedefs

- typedef struct [CFE_TBL_FileDef](#) [CFE_TBL_FileDef_t](#)

12.65.1 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:

12.65.2 Macro Definition Documentation**12.65.2.1 CFE_TBL_FILEDEF**

```
#define CFE_TBL_FILEDEF (
    ObjName,
    TblName,
    Desc,
    Filename )
```

Value:

```
static OS_USED CFE_TBL_FileDef_t CFE_TBL_FileDef = {#ObjName "\0", #
    TblName "\0", #Desc "\0", #Filename "\0", \
        sizeof(ObjName)};
```

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 95 of file cfe_tbl_filedef.h.

12.65.3 Typedef Documentation**12.65.3.1 CFE_TBL_FileDef_t**

```
typedef struct CFE_TBL_FileDef CFE_TBL_FileDef_t
```

12.66 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.

12.66.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.66.2 Macro Definition Documentation

12.66.2.1 [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](#).

12.67 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)

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.

12.67.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.67.2 Macro Definition Documentation

12.67.2.1 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`.

12.67.3 Typedef Documentation

12.67.3.1 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.

12.67.3.2 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 75 of file `cfe_time_api_typedefs.h`.

12.67.4 Enumeration Type Documentation

12.67.4.1 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 60 of file cfe_time_api_typedefs.h.

12.68 cfe/modules/core_api/fsw/inc/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.

12.68.1 Detailed Description

Declarations and prototypes for cfe_time_extern_typedefs module

12.68.2 Typedef Documentation

12.68.2.1 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 249 of file cfe_time_extern_typedefs.h.

12.68.2.2 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 177 of file `cfe_time_extern_typedefs.h`.

12.68.2.3 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 124 of file `cfe_time_extern_typedefs.h`.

12.68.2.4 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 273 of file `cfe_time_extern_typedefs.h`.

12.68.2.5 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 297 of file cfe_time_extern_typedefs.h.

12.68.2.6 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 201 of file cfe_time_extern_typedefs.h.

12.68.2.7 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.

12.68.2.8 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 225 of file cfe_time_extern_typedefs.h.

12.68.3 Enumeration Type Documentation

12.68.3.1 CFE_TIME_AdjustDirection

enum CFE_TIME_AdjustDirection

Label definitions associated with CFE_TIME_AdjustDirection_Enum_t.

Enumerator

CFE_TIME_AdjustDirection_ADD	Add time adjustment.
CFE_TIME_AdjustDirection_SUBTRACT	Subtract time adjustment.

Definition at line 230 of file cfe_time_extern_typedefs.h.

12.68.3.2 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 129 of file cfe_time_extern_typedefs.h.

12.68.3.3 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 60 of file cfe_time_extern_typedefs.h.

12.68.3.4 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 254 of file cfe_time_extern_typedefs.h.

12.68.3.5 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 278 of file cfe_time_extern_typedefs.h.

12.68.3.6 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 182 of file cfe_time_extern_typedefs.h.

12.68.3.7 CFE_TIME_ToneSignalSelect

enum [CFE_TIME_ToneSignalSelect](#)

Label definitions associated with `CFE_TIME_ToneSignalSelect_Enum_t`.

Enumerator

<code>CFE_TIME_ToneSignalSelect_PRIMARY</code>	Primary Source.
<code>CFE_TIME_ToneSignalSelect_REDUNDANT</code>	Redundant Source.

Definition at line 206 of file `cfe_time_extern_typedefs.h`.

12.69 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference

Macros

- `#define CFE_BUILD_NUMBER 136`

Development: Number of development git commits since CFE_BUILD_BASELINE.
- `#define CFE_BUILD_BASELINE "v7.0.0-rc4"`

Development: Reference git tag for build number.
- `#define CFE_MAJOR_VERSION 6`

Major version number.
- `#define CFE_MINOR_VERSION 7`

Minor version number.
- `#define CFE_REVISION 99`

Revision version number. Value of 99 indicates a development version.
- `#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_VERSION_STRING "cFE DEVELOPMENT BUILD " CFE_SRC_VERSION " (Codename: Draco), Last Official Release: cfe v6.7.0"`

Long Build Version String.

12.69.1 Detailed Description

Provide version identifiers for the cFE core. See [Version Numbers](#) for further details.

12.69.2 Macro Definition Documentation

12.69.2.1 CFE_BUILD_BASELINE

```
#define CFE_BUILD_BASELINE "v7.0.0-rc4"
```

Development: Reference git tag for build number.

Definition at line 30 of file cfe_version.h.

12.69.2.2 CFE_BUILD_NUMBER

```
#define CFE_BUILD_NUMBER 136
```

Development: Number of development git commits since CFE_BUILD_BASELINE.

Definition at line 29 of file cfe_version.h.

12.69.2.3 CFE_MAJOR_VERSION

```
#define CFE_MAJOR_VERSION 6
```

Major version number.

Definition at line 33 of file cfe_version.h.

12.69.2.4 CFE_MINOR_VERSION

```
#define CFE_MINOR_VERSION 7
```

Minor version number.

Definition at line 34 of file cfe_version.h.

12.69.2.5 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 44 of file cfe_version.h.

12.69.2.6 CFE_REVISION

```
#define CFE_REVISION 99
```

Revision version number. Value of 99 indicates a development version.

Definition at line 35 of file cfe_version.h.

12.69.2.7 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 55 of file cfe_version.h.

12.69.2.8 CFE_STR

```
#define CFE_STR( x ) CFE_STR_HELPER(x)
```

Expand macro before conversion.

Definition at line 47 of file cfe_version.h.

12.69.2.9 CFE_STR_HELPER

```
#define CFE_STR_HELPER( x ) #x
```

Convert argument to string.

Definition at line 46 of file cfe_version.h.

12.69.2.10 CFE_VERSION_STRING

```
#define CFE_VERSION_STRING "CFE DEVELOPMENT BUILD " CFE_SRC_VERSION " (Codename: Draco), Last  
Official Release: cfe v6.7.0"
```

Long Build Version String.

Long freeform string identifying the build, see [Version Numbers](#) for suggested format for development and official releases.

Definition at line 63 of file cfe_version.h.

12.70 cfe/modules/es/fsw/inc/cfe_es_events.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
ES Write Exception Reset Log Command Request or File Creation Failed Event ID.
- #define `CFE_ES_PERF_STARTCMD_EID` 57

- #define `CFE_ES_PERF_STARTCMD_ERR_EID` 58
 - ES Start Performance Analyzer Data Collection Command Success Event ID.*
- #define `CFE_ES_PERF_STARTCMD_TRIG_ERR_EID` 59
 - ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.*
- #define `CFE_ES_PERF_STOPCMD_EID` 60
 - ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.*
- #define `CFE_ES_PERF_STOPCMD_ERR2_EID` 62
 - ES Stop Performance Analyzer Data Collection Command Request Success Event ID.*
- #define `CFE_ES_PERF_FILTMSKCMD_EID` 63
 - ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.*
- #define `CFE_ES_PERF_FILTMSKERR_EID` 64
 - ES Set Performance Analyzer Filter Mask Command Success Event ID.*
- #define `CFE_ES_PERF_TRIGMSKCMD_EID` 65
 - ES Set Performance Analyzer Filter Mask Command Invalid Index Event ID.*
- #define `CFE_ES_PERF_TRIGMSKERR_EID` 66
 - ES Set Performance Analyzer Trigger Mask Command Success Event ID.*
- #define `CFE_ES_PERF_LOG_ERR_EID` 67
 - ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.*
- #define `CFE_ES_PERF_DATAWRITTEN_EID` 68
 - ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.*
- #define `CFE_ES_CDS_REGISTER_ERR_EID` 69
 - Performance Log Write Success Event ID.*
- #define `CFE_ES_ERR_SYSLOGMODE_EID` 70
 - 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.

12.70.1 Detailed Description

cFE Executive Services Event IDs

12.70.2 Macro Definition Documentation

12.70.2.1 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_events.h`.

12.70.2.2 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_events.h`.

12.70.2.3 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_events.h`.

12.70.2.4 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_events.h`.

12.70.2.5 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_events.h`.

12.70.2.6 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_events.h`.

12.70.2.7 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_events.h.

12.70.2.8 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_events.h.

12.70.2.9 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_events.h.

12.70.2.10 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_events.h.

12.70.2.11 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_events.h.

12.70.2.12 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_events.h.

12.70.2.13 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_events.h.

12.70.2.14 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_events.h.

12.70.2.15 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_events.h.

12.70.2.16 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_events.h.

12.70.2.17 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_events.h.

12.70.2.18 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_events.h.

12.70.2.19 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_events.h.

12.70.2.20 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_events.h.

12.70.2.21 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_events.h.

12.70.2.22 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_events.h.

12.70.2.23 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_events.h.

12.70.2.24 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_events.h.

12.70.2.25 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_events.h.

12.70.2.26 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_events.h.

12.70.2.27 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_events.h`.

12.70.2.28 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_events.h`.

12.70.2.29 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_events.h`.

12.70.2.30 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_events.h.

12.70.2.31 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_events.h.

12.70.2.32 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_events.h.

12.70.2.33 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_events.h.

12.70.2.34 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_events.h.

12.70.2.35 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_events.h.

12.70.2.36 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_events.h.

12.70.2.37 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_events.h.

12.70.2.38 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_events.h.

12.70.2.39 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_events.h.

12.70.2.40 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_events.h.

12.70.2.41 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_events.h.

12.70.2.42 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_events.h.

12.70.2.43 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_P←ERF_DATAWRITTEN_EID](#) event.

Definition at line 674 of file cfe_es_events.h.

12.70.2.44 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_events.h.

12.70.2.45 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_events.h.

12.70.2.46 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_events.h.

12.70.2.47 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_events.h.

12.70.2.48 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_events.h.

12.70.2.49 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_events.h.

12.70.2.50 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_events.h.

12.70.2.51 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_events.h.

12.70.2.52 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_events.h.

12.70.2.53 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_events.h.

12.70.2.54 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_events.h.

12.70.2.55 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_events.h.

12.70.2.56 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_events.h.

12.70.2.57 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_events.h.

12.70.2.58 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_events.h.

12.70.2.59 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_events.h.

12.70.2.60 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_events.h.

12.70.2.61 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_events.h.

12.70.2.62 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_events.h.

12.70.2.63 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_events.h.

12.70.2.64 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_events.h.

12.70.2.65 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_events.h.

12.70.2.66 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_events.h.

12.70.2.67 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_events.h.

12.70.2.68 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_events.h.

12.70.2.69 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_events.h.

12.70.2.70 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_events.h.

12.70.2.71 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_events.h.

12.70.2.72 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_events.h.

12.70.2.73 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_events.h.

12.70.2.74 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_events.h.

12.70.2.75 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_events.h.

12.70.2.76 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_events.h.

12.70.2.77 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_events.h.

12.70.2.78 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_events.h.

12.70.2.79 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_events.h.

12.70.2.80 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_events.h.

12.70.2.81 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_events.h.

12.70.2.82 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_events.h.

12.70.2.83 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_events.h.

12.70.2.84 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_events.h.

12.70.2.85 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_events.h.

12.70.2.86 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_events.h.

12.71 cfe/modules/es/fsw/inc/cfe_es_msg.h File Reference

```
#include "common_types.h"
#include "cfe_msg_hdr.h"
#include "cfe_es_extern_typedefs.h"
```

Data Structures

- struct [CFE_ES_NoArgsCmd](#)
Generic "no arguments" command.
- struct [CFE_ES_RestartCmd_Payload](#)
Restart cFE Command Payload.
- struct [CFE_ES_RestartCmd](#)
Restart cFE Command.
- struct [CFE_ES_FileNameCmd_Payload](#)
Generic file name command payload.
- struct [CFE_ES_FileNameCmd](#)
Generic file name command.
- struct [CFE_ES_OverWriteSysLogCmd_Payload](#)
Overwrite/Discard System Log Configuration Command Payload.
- struct [CFE_ES_OverWriteSysLogCmd](#)
Overwrite/Discard System Log Configuration Command Payload.
- struct [CFE_ES_StartAppCmd_Payload](#)
Start Application Command Payload.
- struct [CFE_ES_StartApp](#)
Start Application Command.
- struct [CFE_ES_AppNameCmd_Payload](#)
Generic application name command payload.

- struct [CFE_ES_AppNameCmd](#)
Generic application name command.
- struct [CFE_ES_AppReloadCmd_Payload](#)
Reload Application Command Payload.
- struct [CFE_ES_ReloadAppCmd](#)
Reload Application Command.
- struct [CFE_ES_SetMaxPRCountCmd_Payload](#)
Set Maximum Processor Reset Count Command Payload.
- struct [CFE_ES_SetMaxPRCountCmd](#)
Set Maximum Processor Reset Count Command.
- struct [CFE_ES_DeleteCDSCmd_Payload](#)
Delete Critical Data Store Command Payload.
- struct [CFE_ES_DeleteCDSCmd](#)
Delete Critical Data Store Command.
- struct [CFE_ES_StartPerfCmd_Payload](#)
Start Performance Analyzer Command Payload.
- struct [CFE_ES_StartPerfDataCmd](#)
Start Performance Analyzer Command.
- struct [CFE_ES_StopPerfCmd_Payload](#)
Stop Performance Analyzer Command Payload.
- struct [CFE_ES_StopPerfDataCmd](#)
Stop Performance Analyzer Command.
- struct [CFE_ES_SetPerfFilterMaskCmd_Payload](#)
Set Performance Analyzer Filter Mask Command Payload.
- struct [CFE_ES_SetPerfFilterMaskCmd](#)
Set Performance Analyzer Filter Mask Command.
- struct [CFE_ES_SetPerfTrigMaskCmd_Payload](#)
Set Performance Analyzer Trigger Mask Command Payload.
- struct [CFE_ES_SetPerfTriggerMaskCmd](#)
Set Performance Analyzer Trigger Mask Command.
- struct [CFE_ES_SendMemPoolStatsCmd_Payload](#)
Send Memory Pool Statistics Command Payload.
- struct [CFE_ES_SendMemPoolStatsCmd](#)
Send Memory Pool Statistics Command.
- struct [CFE_ES_DumpCDSRegistryCmd_Payload](#)
Dump CDS Registry Command Payload.
- struct [CFE_ES_DumpCDSRegistryCmd](#)
Dump CDS Registry Command.
- struct [CFE_ES_OneAppTlm_Payload](#)
- struct [CFE_ES_OneAppTlm](#)
- struct [CFE_ES_PoolStatsTlm_Payload](#)
- struct [CFE_ES_MemStatsTlm](#)
- struct [CFE_ES_HousekeepingTlm_Payload](#)
- struct [CFE_ES_HousekeepingTlm](#)

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_SYSLOG_CC 10
- #define CFE_ES_WRITE_SYSLOG_CC 11
- #define CFE_ES_CLEAR_ER_LOG_CC 12
- #define CFE_ES_WRITE_ER_LOG_CC 13
- #define CFE_ES_START_PERF_DATA_CC 14
- #define CFE_ES_STOP_PERF_DATA_CC 15
- #define CFE_ES_SET_PERF_FILTER_MASK_CC 16
- #define CFE_ES_SET_PERF_TRIGGER_MASK_CC 17
- #define CFE_ES_OVER_WRITE_SYSLOG_CC 18
- #define CFE_ES_RESET_PR_COUNT_CC 19
- #define CFE_ES_SET_MAX_PR_COUNT_CC 20
- #define CFE_ES_DELETE_CDS_CC 21
- #define CFE_ES_SEND_MEM_POOL_STATS_CC 22
- #define CFE_ES_DUMP_CDS_REGISTRY_CC 23
- #define CFE_ES_QUERY_ALL_TASKS_CC 24

Typedefs

- typedef struct CFE_ES_NoArgsCmd CFE_ES_NoArgsCmd_t

Generic "no arguments" command.
- typedef CFE_ES_NoArgsCmd_t CFE_ES_NoopCmd_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCountersCmd_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSysLogCmd_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLogCmd_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCountCmd_t
- typedef struct CFE_ES_RestartCmd_Payload CFE_ES_RestartCmd_Payload_t

Restart cFE Command Payload.
- typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t

Restart cFE Command.
- typedef struct CFE_ES_FileNameCmd_Payload CFE_ES_FileNameCmd_Payload_t

Generic file name command payload.
- typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t

Generic file name command.
- typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllCmd_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasksCmd_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSysLogCmd_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLogCmd_t
- typedef struct CFE_ES_OverWriteSysLogCmd_Payload CFE_ES_OverWriteSysLogCmd_Payload_t

Overwrite/Discard System Log Configuration Command Payload.
- typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_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_StartApp CFE_ES_StartAppCmd_t**
Start Application Command.
- **typedef struct CFE_ES_AppNameCmd_Payload CFE_ES_AppNameCmd_Payload_t**
Generic application name command payload.
- **typedef struct CFE_ES_AppNameCmd CFE_ES_AppNameCmd_t**
Generic application name command.
- **typedef CFE_ES_AppNameCmd_t CFE_ES_StopAppCmd_t**
- **typedef CFE_ES_AppNameCmd_t CFE_ES_RestartAppCmd_t**
- **typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOneCmd_t**
- **typedef struct CFE_ES_AppReloadCmd_Payload CFE_ES_AppReloadCmd_Payload_t**
Reload Application Command Payload.
- **typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t**
Reload Application Command.
- **typedef struct CFE_ES_SetMaxPRCountCmd_Payload CFE_ES_SetMaxPRCountCmd_Payload_t**
Set Maximum Processor Reset Count Command Payload.
- **typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t**
Set Maximum Processor Reset Count Command.
- **typedef struct CFE_ES_DeleteCDSCmd_Payload CFE_ES_DeleteCDSCmd_Payload_t**
Delete Critical Data Store Command Payload.
- **typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t**
Delete Critical Data Store Command.
- **typedef struct CFE_ES_StartPerfCmd_Payload CFE_ES_StartPerfCmd_Payload_t**
Start Performance Analyzer Command Payload.
- **typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t**
Start Performance Analyzer Command.
- **typedef struct CFE_ES_StopPerfCmd_Payload CFE_ES_StopPerfCmd_Payload_t**
Stop Performance Analyzer Command Payload.
- **typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t**
Stop Performance Analyzer Command.
- **typedef struct CFE_ES_SetPerfFilterMaskCmd_Payload CFE_ES_SetPerfFilterMaskCmd_Payload_t**
Set Performance Analyzer Filter Mask Command Payload.
- **typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t**
Set Performance Analyzer Filter Mask Command.
- **typedef struct CFE_ES_SetPerfTrigMaskCmd_Payload CFE_ES_SetPerfTrigMaskCmd_Payload_t**
Set Performance Analyzer Trigger Mask Command Payload.
- **typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t**
Set Performance Analyzer Trigger Mask Command.
- **typedef struct CFE_ES_SendMemPoolStatsCmd_Payload CFE_ES_SendMemPoolStatsCmd_Payload_t**
Send Memory Pool Statistics Command Payload.
- **typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t**
Send Memory Pool Statistics Command.
- **typedef struct CFE_ES_DumpCDSRegistryCmd_Payload CFE_ES_DumpCDSRegistryCmd_Payload_t**
Dump CDS Registry Command Payload.
- **typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t**

Dump CDS Registry Command.

- `typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t`
- `typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t`
- `typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t`
- `typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t`
- `typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t`
- `typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t`

12.71.1 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:

12.71.2 Macro Definition Documentation

12.71.2.1 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_SYSLOG_CC`, `CFE_ES_WRITE_SYSLOG_CC`, `CFE_ES_WRITE_ER_LOG_CC`

Definition at line 550 of file `cfe_es_msg.h`.

12.71.2.2 CFE_ES_CLEAR_SYSLOG_CC

```
#define CFE_ES_CLEAR_SYSLOG_CC 10
```

Name Clear Executive Services System Log

Description

This command clears the contents of the Executive Services System Log.

Command Mnemonic(s) \$sc_\$cpu_ES_ClearSysLog

Command Structure

[CFE_ES_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_SYSLOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#), [CFE_ES_WRITE_ER_LOG_CC](#), [CFE_ES_OVER_WRITE_SYSLOG_CC](#)

Definition at line 473 of file cfe_es_msg.h.

12.71.2.3 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 919 of file `cfe_es_msg.h`.

12.71.2.4 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 1000 of file `cfe_es_msg.h`.

12.71.2.5 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 83 of file [cfe_es_msg.h](#).

12.71.2.6 CFE_ES_OVER_WRITE_SYSLOG_CC

```
#define CFE_ES_OVER_WRITE_SYSLOG_CC 18
```

Name Set Executive Services System Log Mode to Discard/Overwrite

Description

This command allows the user to configure the Executive Services to either discard new System Log messages when it is full or to overwrite the oldest messages.

Command Mnemonic(s) \$sc_\$cpu_ES_OverwriteSysLogMode

Command Structure

[CFE_ES_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_SYSLOG_CC](#), [CFE_ES_WRITE_SYSLOG_CC](#)

Definition at line 802 of file cfe_es_msg.h.

12.71.2.7 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 438 of file cfe_es_msg.h.

12.71.2.8 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_TASKLOG_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 1042 of file `cfe_es_msg.h`.

12.71.2.9 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 396 of file `cfe_es_msg.h`.

12.71.2.10 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 360 of file `cfe_es_msg.h`.

12.71.2.11 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 120 of file `cfe_es_msg.h`.

12.71.2.12 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_ResetPRCntr

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 839 of file cfe_es_msg.h.

12.71.2.13 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 314 of file `cfe_es_msg.h`.

12.71.2.14 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 172 of file `cfe_es_msg.h`.

12.71.2.15 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 958 of file [cfe_es_msg.h](#).

12.71.2.16 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 877 of file cfe_es_msg.h.

12.71.2.17 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 725 of file `cfe_es_msg.h`.

12.71.2.18 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 762 of file cfe_es_msg.h.

12.71.2.19 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_RESTART](#) (1)
- The Operating System was unable to load the specified application file

Evidence of failure may be found in the following telemetry:

- [\\$sc_\\$cpu_ES_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous although system resources could be taxed beyond their limits with the starting of erroneous or invalid applications.

See also

[CFE_ES_STOP_APP_CC](#), [CFE_ES_RESTART_APP_CC](#), [CFE_ES_RELOAD_APP_CC](#)

Definition at line 215 of file `cfe_es_msg.h`.

12.71.2.20 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 638 of file cfe_es_msg.h.

12.71.2.21 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 268 of file cfe_es_msg.h.

12.71.2.22 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 688 of file cfe_es_msg.h.

12.71.2.23 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_SYSLOG_CC](#), [CFE_ES_WRITE_SYSLOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#)

Definition at line 593 of file `cfe_es_msg.h`.

12.71.2.24 CFE_ES_WRITE_SYSLOG_CC

```
#define CFE_ES_WRITE_SYSLOG_CC 11
```

Name Writes contents of Executive Services System Log to a File

Description

This command causes the contents of the Executive Services System Log to be written to a log file.

Command Mnemonic(s) \$sc_\$cpu_ES_WriteSysLog2File

Command Structure

[CFE_ES_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_SYSLOG_CC](#), [CFE_ES_CLEAR_ER_LOG_CC](#), [CFE_ES_WRITE_ER_LOG_CC](#), [CFE_ES_OVER_WRITE_SYSLOG_CC](#)

Definition at line 516 of file `cfe_es_msg.h`.

12.71.3 Typedef Documentation

12.71.3.1 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](#)

12.71.3.2 CFE_ES_AppNameCmd_t

```
typedef struct CFE_ES_AppNameCmd CFE_ES_AppNameCmd_t
```

Generic application name command.

12.71.3.3 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](#)

12.71.3.4 CFE_ES_ClearERLogCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLogCmd_t
```

Definition at line 1074 of file cfe_es_msg.h.

12.71.3.5 CFE_ES_ClearSysLogCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSysLogCmd_t
```

Definition at line 1073 of file cfe_es_msg.h.

12.71.3.6 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](#)

12.71.3.7 CFE_ES_DeleteCDSCmd_t

```
typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t
```

Delete Critical Data Store Command.

12.71.3.8 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](#)

12.71.3.9 CFE_ES_DumpCDSRegistryCmd_t

```
typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t
```

Dump CDS Registry Command.

12.71.3.10 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_SYSLOG_CC](#), and [CFE_ES_WRITE_ER_LOG_CC](#)

12.71.3.11 CFE_ES_FileNameCmd_t

```
typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t
```

Generic file name command.

12.71.3.12 CFE_ES_HousekeepingTlm_Payload_t

```
typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t
```

Name Executive Services Housekeeping Packet

12.71.3.13 CFE_ES_HousekeepingTlm_t

```
typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t
```

12.71.3.14 CFE_ES_MemStatsTlm_t

```
typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t
```

12.71.3.15 CFE_ES_NoArgsCmd_t

```
typedef struct CFE_ES_NoArgsCmd CFE_ES_NoArgsCmd_t
```

Generic "no arguments" command.

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see [CFE_ES_NOOP_CC](#))
3. The Reset Counters Command (For details, see [CFE_ES_RESET_COUNTERS_CC](#))

12.71.3.16 CFE_ES_NoopCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_NoopCmd_t
```

Definition at line 1071 of file cfe_es_msg.h.

12.71.3.17 CFE_ES_OneAppTlm_Payload_t

```
typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t
```

Name Single Application Information Packet

12.71.3.18 CFE_ES_OneAppTlm_t

```
typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t
```

12.71.3.19 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_SYSLOG_CC](#)

12.71.3.20 CFE_ES_OverWriteSysLogCmd_t

```
typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_t
```

Overwrite/Discard System Log Configuration Command Payload.

12.71.3.21 CFE_ES_PoolStatsTlm_Payload_t

```
typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t
```

Name Memory Pool Statistics Packet

12.71.3.22 CFE_ES_QueryAllCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllCmd_t
```

Definition at line 1125 of file cfe_es_msg.h.

12.71.3.23 CFE_ES_QueryAllTasksCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasksCmd_t
```

Definition at line 1126 of file cfe_es_msg.h.

12.71.3.24 CFE_ES_QueryOneCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOneCmd_t
```

Definition at line 1211 of file cfe_es_msg.h.

12.71.3.25 CFE_ES_ReloadAppCmd_t

```
typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t
```

Reload Application Command.

12.71.3.26 CFE_ES_ResetCountersCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCountersCmd_t
```

Definition at line 1072 of file cfe_es_msg.h.

12.71.3.27 CFE_ES_ResetPRCountCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCountCmd_t
```

Definition at line 1075 of file cfe_es_msg.h.

12.71.3.28 CFE_ES_RestartAppCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_RestartAppCmd_t
```

Definition at line 1210 of file cfe_es_msg.h.

12.71.3.29 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](#)

12.71.3.30 CFE_ES_RestartCmd_t

```
typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t
```

Restart cFE Command.

12.71.3.31 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](#)

12.71.3.32 CFE_ES_SendMemPoolStatsCmd_t

```
typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t
```

Send Memory Pool Statistics Command.

12.71.3.33 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](#)

12.71.3.34 CFE_ES_SetMaxPRCountCmd_t

```
typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t
```

Set Maximum Processor Reset Count Command.

12.71.3.35 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](#)

12.71.3.36 CFE_ES_SetPerfFilterMaskCmd_t

```
typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t
```

Set Performance Analyzer Filter Mask Command.

12.71.3.37 CFE_ES_SetPerfTriggerMaskCmd_t

```
typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t
```

Set Performance Analyzer Trigger Mask Command.

12.71.3.38 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](#)

12.71.3.39 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](#)

12.71.3.40 CFE_ES_StartAppCmd_t

```
typedef struct CFE_ES_StartApp CFE_ES_StartAppCmd_t
```

Start Application Command.

12.71.3.41 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](#)

12.71.3.42 CFE_ES_StartPerfDataCmd_t

```
typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t
```

Start Performance Analyzer Command.

12.71.3.43 CFE_ES_StopAppCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_StopAppCmd_t
```

Definition at line 1209 of file cfe_es_msg.h.

12.71.3.44 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](#)

12.71.3.45 CFE_ES_StopPerfDataCmd_t

```
typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t
```

Stop Performance Analyzer Command.

12.71.3.46 CFE_ES_WriteERLogCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLogCmd_t
```

Definition at line 1128 of file cfe_es_msg.h.

12.71.3.47 CFE_ES_WriteSysLogCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSysLogCmd_t
```

Definition at line 1127 of file cfe_es_msg.h.

12.72 cfe/modules/evs/fsw/inc/cfe_evs_events.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_EVTIDMOREGS_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
EVS Add Filter Command Max Filters Exceeded Event ID.
- #define `CFE_EVS_ERR_WRDATFILE_EID` 12
EVS Write Application Data Command Write Data Failure Event ID.
- #define `CFE_EVS_ERR_CRDATFILE_EID` 13
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_ENAAPPEVT_EID` 25
EVS Enable App Events Command Success Event ID.

- #define [CFE_EVS_DISAPPEVT_EID](#) 26
EVS Disable App Events Command Success Event ID.
- #define [CFE_EVS_RSTEVTCNT_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.

12.72.1 Detailed Description

cFE Event Services Event IDs

12.72.2 Macro Definition Documentation

12.72.2.1 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 356 of file cfe_evs_events.h.

12.72.2.2 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 367 of file cfe_evs_events.h.

12.72.2.3 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 290 of file cfe_evs_events.h.

12.72.2.4 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 312 of file cfe_evs_events.h.

12.72.2.5 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 257 of file cfe_evs_events.h.

12.72.2.6 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 235 of file cfe_evs_events.h.

12.72.2.7 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 301 of file cfe_evs_events.h.

12.72.2.8 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 279 of file cfe_evs_events.h.

12.72.2.9 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 246 of file cfe_evs_events.h.

12.72.2.10 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 224 of file cfe_evs_events.h.

12.72.2.11 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_events.h.

12.72.2.12 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 191 of file cfe_evs_events.h.

12.72.2.13 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_events.h.

12.72.2.14 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_events.h.

12.72.2.15 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_events.h.

12.72.2.16 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_events.h.

12.72.2.17 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_events.h.

12.72.2.18 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 435 of file cfe_evs_events.h.

12.72.2.19 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 424 of file cfe_evs_events.h.

12.72.2.20 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_events.h.

12.72.2.21 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_events.h.

12.72.2.22 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_events.h.

12.72.2.23 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 446 of file cfe_evs_events.h.

12.72.2.24 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_events.h.

12.72.2.25 CFE_EVS_ERR_WRLOGFILE_EID

```
#define CFE_EVS_ERR_WRLOGFILE_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_events.h.

12.72.2.26 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 401 of file cfe_evs_events.h.

12.72.2.27 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 457 of file cfe_evs_events.h.

12.72.2.28 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 468 of file cfe_evs_events.h.

12.72.2.29 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 412 of file cfe_evs_events.h.

12.72.2.30 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_events.h.

12.72.2.31 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 345 of file cfe_evs_events.h.

12.72.2.32 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 202 of file cfe_evs_events.h.

12.72.2.33 CFE_EVS_RSTEVCNT_EID

```
#define CFE_EVS_RSTEVCNT_EID 27
```

EVS Reset App Event Counter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset App Event Counter Command](#) success.

Definition at line 323 of file cfe_evs_events.h.

12.72.2.34 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 334 of file cfe_evs_events.h.

12.72.2.35 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 268 of file cfe_evs_events.h.

12.72.2.36 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 213 of file cfe_evs_events.h.

12.72.2.37 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_events.h.

12.72.2.38 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 378 of file cfe_evs_events.h.

12.72.2.39 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 389 of file cfe_evs_events.h.

12.73 cfe/modules/evs/fsw/inc/cfe_evs_msg.h File Reference

```
#include "common_types.h"
#include "cfe_msg_hdr.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_es_extern_typedefs.h"
```

Data Structures

- struct [CFE_EVS_NoArgsCmd](#)
Command with no additional arguments.
- struct [CFE_EVS_LogFileCmd_Payload](#)
Write Event Log to File Command Payload.
- struct [CFE_EVS_WriteLogFileCmd](#)
Write Event Log to File Command.
- struct [CFE_EVS_AppDataCmd_Payload](#)
Write Event Services Application Information to File Command Payload.
- struct [CFE_EVS_WriteAppDataFileCmd](#)
Write Event Services Application Information to File Command.
- struct [CFE_EVS_SetLogMode_Payload](#)
Set Log Mode Command Payload.
- struct [CFE_EVS_SetLogModeCmd](#)
Set Log Mode Command.
- struct [CFE_EVS_SetEventFormatCode_Payload](#)
Set Event Format Mode Command Payload.
- struct [CFE_EVS_SetEventFormatModeCmd](#)
Set Event Format Mode Command.
- struct [CFE_EVS_BitMaskCmd_Payload](#)
Generic Bitmask Command Payload.
- struct [CFE_EVS_BitMaskCmd](#)
Generic Bitmask Command.
- struct [CFE_EVS_AppNameCmd_Payload](#)
Generic App Name Command Payload.
- struct [CFE_EVS_AppNameCmd](#)
Generic App Name Command.
- struct [CFE_EVS_AppNameEventIDCmd_Payload](#)
Generic App Name and Event ID Command Payload.
- struct [CFE_EVS_AppNameEventIDCmd](#)
Generic App Name and Event ID Command.
- struct [CFE_EVS_AppNameBitMaskCmd_Payload](#)
Generic App Name and Bitmask Command Payload.
- struct [CFE_EVS_AppNameBitMaskCmd](#)
Generic App Name and Bitmask Command.
- struct [CFE_EVS_AppNameEventIDMaskCmd_Payload](#)
Generic App Name, Event ID, Mask Command Payload.
- struct [CFE_EVS_AppNameEventIDMaskCmd](#)
Generic App Name, Event ID, Mask Command.
- struct [CFE_EVS_AppTlmData](#)
- struct [CFE_EVS_HousekeepingTlm_Payload](#)
- struct [CFE_EVS_HousekeepingTlm](#)
- struct [CFE_EVS_PacketID](#)
- struct [CFE_EVS_LongEventTlm_Payload](#)
- struct [CFE_EVS_ShortEventTlm_Payload](#)
- struct [CFE_EVS_LongEventTlm](#)
- struct [CFE_EVS_ShortEventTlm](#)

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

Event Services Command Codes

- #define CFE_EVS_NOOP_CC 0
- #define CFE_EVS_RESET_COUNTERS_CC 1
- #define CFE_EVS_ENABLE_EVENT_TYPE_CC 2
- #define CFE_EVS_DISABLE_EVENT_TYPE_CC 3
- #define CFE_EVS_SET_EVENT_FORMAT_MODE_CC 4
- #define CFE_EVS_ENABLE_APP_EVENT_TYPE_CC 5
- #define CFE_EVS_DISABLE_APP_EVENT_TYPE_CC 6
- #define CFE_EVS_ENABLE_APP_EVENTS_CC 7
- #define CFE_EVS_DISABLE_APP_EVENTS_CC 8
- #define CFE_EVS_RESET_APP_COUNTER_CC 9
- #define CFE_EVS_SET_FILTER_CC 10
- #define CFE_EVS_ENABLE_PORTS_CC 11
- #define CFE_EVS_DISABLE_PORTS_CC 12
- #define CFE_EVS_RESET_FILTER_CC 13
- #define CFE_EVS_RESET_ALL_FILTERS_CC 14
- #define CFE_EVS_ADD_EVENT_FILTER_CC 15
- #define CFE_EVS_DELETE_EVENT_FILTER_CC 16
- #define CFE_EVS_WRITE_APP_DATA_FILE_CC 17
- #define CFE_EVS_WRITE_LOG_DATA_FILE_CC 18
- #define CFE_EVS_SET_LOG_MODE_CC 19
- #define CFE_EVS_CLEAR_LOG_CC 20

TypeDefs

- typedef struct CFE_EVS_NoArgsCmd CFE_EVS_NoArgsCmd_t
Command with no additional arguments.
- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_NoopCmd_t
- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCountersCmd_t
- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLogCmd_t
- typedef struct CFE_EVS_LogFileCmd_Payload CFE_EVS_LogFileCmd_Payload_t
Write Event Log to File Command Payload.
- typedef struct CFE_EVS_WriteLogFileCmd CFE_EVS_WriteLogFileCmd_t
Write Event Log to File Command.
- typedef struct CFE_EVS_AppDataCmd_Payload CFE_EVS_AppDataCmd_Payload_t
Write Event Services Application Information to File Command Payload.
- typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd_t
Write Event Services Application Information to File Command.
- typedef struct CFE_EVS_SetLogMode_Payload CFE_EVS_SetLogMode_Payload_t
Set Log Mode Command Payload.

- **typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t**
Set Log Mode Command.
- **typedef struct CFE_EVS_SetEventFormatCode_Payload CFE_EVS_SetEventFormatMode_Payload_t**
Set Event Format Mode Command Payload.
- **typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t**
Set Event Format Mode Command.
- **typedef struct CFE_EVS_BitMaskCmd_Payload CFE_EVS_BitMaskCmd_Payload_t**
Generic Bitmask Command Payload.
- **typedef struct CFE_EVS_BitMaskCmd CFE_EVS_BitMaskCmd_t**
Generic Bitmask Command.
- **typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePortsCmd_t**
- **typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePortsCmd_t**
- **typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventTypeCmd_t**
- **typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventTypeCmd_t**
- **typedef struct CFE_EVS_AppNameCmd_Payload CFE_EVS_AppNameCmd_Payload_t**
Generic App Name Command Payload.
- **typedef struct CFE_EVS_AppNameCmd CFE_EVS_AppNameCmd_t**
Generic App Name Command.
- **typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEventsCmd_t**
- **typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEventsCmd_t**
- **typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounterCmd_t**
- **typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFiltersCmd_t**
- **typedef struct CFE_EVS_AppNameEventIDCmd_Payload CFE_EVS_AppNameEventIDCmd_Payload_t**
Generic App Name and Event ID Command Payload.
- **typedef struct CFE_EVS_AppNameEventIDCmd CFE_EVS_AppNameEventIDCmd_t**
Generic App Name and Event ID Command.
- **typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilterCmd_t**
- **typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilterCmd_t**
- **typedef struct CFE_EVS_AppNameBitMaskCmd_Payload CFE_EVS_AppNameBitMaskCmd_Payload_t**
Generic App Name and Bitmask Command Payload.
- **typedef struct CFE_EVS_AppNameBitMaskCmd CFE_EVS_AppNameBitMaskCmd_t**
Generic App Name and Bitmask Command.
- **typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventTypeCmd_t**
- **typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventTypeCmd_t**
- **typedef struct CFE_EVS_AppNameEventIDMaskCmd_Payload CFE_EVS_AppNameEventIDMaskCmd_t**
Generic App Name, Event ID, Mask Command Payload.
- **typedef struct CFE_EVS_AppNameEventIDMaskCmd CFE_EVS_AppNameEventIDMaskCmd_t**
Generic App Name, Event ID, Mask Command.
- **typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_AddEventFilterCmd_t**
- **typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilterCmd_t**
- **typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t**
- **typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t**
- **typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_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**
- **typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t**
- **typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t**

12.73.1 Detailed Description

Title: Event Services Message definition header file Header File

Purpose: Unit specification for Event services command codes and data structures.

Design Notes:

References: Flight Software Branch C Coding Standard Version 1.0a

12.73.2 Macro Definition Documentation

12.73.2.1 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 705 of file cfe_evs_msg.h.

12.73.2.2 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 885 of file cfe_evs_msg.h.

12.73.2.3 CFE_EVS_CRITICAL_BIT

```
#define CFE_EVS_CRITICAL_BIT 0x0008
```

Definition at line 892 of file cfe_evs_msg.h.

12.73.2.4 CFE_EVS_DEBUG_BIT

```
#define CFE_EVS_DEBUG_BIT 0x0001
```

Definition at line 889 of file cfe_evs_msg.h.

12.73.2.5 CFE_EVS_DELETE_EVENT_FILTER_CC

```
#define CFE_EVS_DELETE_EVENT_FILTER_CC 16
```

Name Delete Application Event Filter

Description

This command removes the given filter for the given application identifier and event identifier. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) \$sc_\$cpu_EVS_DelEvtFltr

Command Structure

[CFE_EVS_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 740 of file cfe_evs_msg.h.

12.73.2.6 CFE_EVS_DISABLE_APP_EVENT_TYPE_CC

```
#define CFE_EVS_DISABLE_APP_EVENT_TYPE_CC 6
```

Name Disable Application Event Type

Description

This command disables the command specified event type for the command specified application, preventing the application from sending event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) \$sc_\$cpu_EVS_DisAppEvtType, \$sc_\$cpu_EVS_DisAppEvtTypeMask

Command Structure

[CFE_EVS_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 365 of file cfe_evs_msg.h.

12.73.2.7 CFE_EVS_DISABLE_APP_EVENTS_CC

```
#define CFE_EVS_DISABLE_APP_EVENTS_CC 8
```

Name Disable Event Services for an Application

Description

This command disables the command specified application from sending events through Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) \$sc_\$cpu_EVS_DisAppEvGen

Command Structure

[CFE_EVS_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 443 of file `cfe_evs_msg.h`.

12.73.2.8 CFE_EVS_DISABLE_EVENT_TYPE_CC

```
#define CFE_EVS_DISABLE_EVENT_TYPE_CC 3
```

Name Disable Event Type

Description

This command disables the command specified Event Type preventing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global disable of a particular event type, it applies to all applications.

Command Mnemonic(s) \$sc_\$cpu_EVS_DisEventType, \$sc_\$cpu_EVS_DisEventTypeMask

Command Structure

[CFE_EVS_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 213 of file `cfe_evs_msg.h`.

12.73.2.9 CFE_EVS_DISABLE_PORTS_CC

```
#define CFE_EVS_DISABLE_PORTS_CC 12
```

Name Disable Event Services Output Ports

Description

This command disables the specified port from outputting event messages.

Command Mnemonic(s) \$sc_\$cpu_EVS_DisPort, \$sc_\$cpu_EVS_DisPortMask

Command Structure

CFE_EVS_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 599 of file cfe_evs_msg.h.

12.73.2.10 CFE_EVS_ENABLE_APP_EVENT_TYPE_CC

```
#define CFE_EVS_ENABLE_APP_EVENT_TYPE_CC 5
```

Name Enable Application Event Type

Description

This command enables the command specified event type for the command specified application, allowing the application to send event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) \$sc_\$cpu_EVS_EnaAppEvtType, \$sc_\$cpu_EVS_EnaAppEvtTypeMask

Command Structure

[CFE_EVS_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_ENAAPPEVTTYPE_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 312 of file cfe_evs_msg.h.

12.73.2.11 CFE_EVS_ENABLE_APP_EVENTS_CC

```
#define CFE_EVS_ENABLE_APP_EVENTS_CC 7
```

Name Enable Event Services for an Application

Description

This command enables the command specified application to send events through the Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) \$sc_\$cpu_EVS_EnaAppEvGen

Command Structure

[CFE_EVS_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_ENAAPPEVT_EID](#) debug event message
- The setting of the Active Flag in The Active Flag in EVS App Data File

Error Conditions

This command may fail for the following reason(s):

- 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 404 of file `cfe_evs_msg.h`.

12.73.2.12 CFE_EVS_ENABLE_EVENT_TYPE_CC

```
#define CFE_EVS_ENABLE_EVENT_TYPE_CC 2
```

Name Enable Event Type

Description

This command enables the command specified Event Type allowing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global enable of a particular event type, it applies to all applications.

Command Mnemonic(s) \$sc_\$cpu_EVS_EnaEventType, \$sc_\$cpu_EVS_EnaEventTypeMask

Command Structure

[CFE_EVS_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 164 of file cfe_evs_msg.h.

12.73.2.13 CFE_EVS_ENABLE_PORTS_CC

```
#define CFE_EVS_ENABLE_PORTS_CC 11
```

Name Enable Event Services Output Ports

Description

This command enables the command specified port to output event messages

Command Mnemonic(s) \$sc_\$cpu_EVS_EnaPort, \$sc_\$cpu_EVS_EnaPortMask

Command Structure

CFE_EVS_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 560 of file cfe_evs_msg.h.

12.73.2.14 CFE_EVS_ERROR_BIT

```
#define CFE_EVS_ERROR_BIT 0x0004
```

Definition at line 891 of file cfe_evs_msg.h.

12.73.2.15 CFE_EVS_INFORMATION_BIT

```
#define CFE_EVS_INFORMATION_BIT 0x0002
```

Definition at line 890 of file cfe_evs_msg.h.

12.73.2.16 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 77 of file cfe_evs_msg.h.

12.73.2.17 CFE_EVS_PORT1_BIT

```
#define CFE_EVS_PORT1_BIT 0x0001
```

Definition at line 895 of file cfe_evs_msg.h.

12.73.2.18 CFE_EVS_PORT2_BIT

```
#define CFE_EVS_PORT2_BIT 0x0002
```

Definition at line 896 of file cfe_evs_msg.h.

12.73.2.19 CFE_EVS_PORT3_BIT

```
#define CFE_EVS_PORT3_BIT 0x0004
```

Definition at line 897 of file cfe_evs_msg.h.

12.73.2.20 CFE_EVS_PORT4_BIT

```
#define CFE_EVS_PORT4_BIT 0x0008
```

Definition at line 898 of file cfe_evs_msg.h.

12.73.2.21 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 669 of file `cfe_evs_msg.h`.

12.73.2.22 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 479 of file `cfe_evs_msg.h`.

12.73.2.23 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 116 of file cfe_evs_msg.h.

12.73.2.24 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 635 of file cfe_evs_msg.h.

12.73.2.25 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 260 of file cfe_evs_msg.h.

12.73.2.26 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 521 of file `cfe_evs_msg.h`.

12.73.2.27 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 850 of file cfe_evs_msg.h.

12.73.2.28 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 779 of file `cfe_evs_msg.h`.

12.73.2.29 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 814 of file `cfe_evs_msg.h`.

12.73.3 Typedef Documentation

12.73.3.1 `CFE_EVS_AddEventFilterCmd_t`

```
typedef CFE\_EVS\_AppNameEventIDMaskCmd\_t CFE_EVS_AddEventFilterCmd_t
```

Definition at line 1151 of file `cfe_evs_msg.h`.

12.73.3.2 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](#)

12.73.3.3 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](#)

12.73.3.4 CFE_EVS_AppNameBitMaskCmd_t

```
typedef struct CFE_EVS_AppNameBitMaskCmd CFE_EVS_AppNameBitMaskCmd_t
```

Generic App Name and Bitmask Command.

12.73.3.5 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](#)

12.73.3.6 CFE_EVS_AppNameCmd_t

```
typedef struct CFE_EVS_AppNameCmd CFE_EVS_AppNameCmd_t
```

Generic App Name Command.

12.73.3.7 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](#)

12.73.3.8 CFE_EVS_AppNameEventIDCmd_t

```
typedef struct CFE_EVS_AppNameEventIDCmd CFE_EVS_AppNameEventIDCmd_t
```

Generic App Name and Event ID Command.

12.73.3.9 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](#)

12.73.3.10 CFE_EVS_AppNameEventIDMaskCmd_t

```
typedef struct CFE_EVS_AppNameEventIDMaskCmd CFE_EVS_AppNameEventIDMaskCmd_t
```

Generic App Name, Event ID, Mask Command.

12.73.3.11 CFE_EVS_AppTlmData_t

```
typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t
```

12.73.3.12 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](#)

12.73.3.13 CFE_EVS_BitMaskCmd_t

```
typedef struct CFE_EVS_BitMaskCmd CFE_EVS_BitMaskCmd_t
```

Generic Bitmask Command.

12.73.3.14 CFE_EVS_ClearLogCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLogCmd_t
```

Definition at line 917 of file cfe_evs_msg.h.

12.73.3.15 CFE_EVS_DeleteEventFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilterCmd_t
```

Definition at line 1091 of file cfe_evs_msg.h.

12.73.3.16 CFE_EVS_DisableAppEventsCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEventsCmd_t
```

Definition at line 1060 of file cfe_evs_msg.h.

12.73.3.17 CFE_EVS_DisableAppEventTypeCmd_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventTypeCmd_t
```

Definition at line 1121 of file cfe_evs_msg.h.

12.73.3.18 CFE_EVS_DisableEventTypeCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventTypeCmd_t
```

Definition at line 1031 of file cfe_evs_msg.h.

12.73.3.19 CFE_EVS_DisablePortsCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePortsCmd_t
```

Definition at line 1029 of file cfe_evs_msg.h.

12.73.3.20 CFE_EVS_EnableAppEventsCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEventsCmd_t
```

Definition at line 1059 of file cfe_evs_msg.h.

12.73.3.21 CFE_EVS_EnableAppEventTypeCmd_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventTypeCmd_t
```

Definition at line 1120 of file cfe_evs_msg.h.

12.73.3.22 CFE_EVS_EnableEventTypeCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventTypeCmd_t
```

Definition at line 1030 of file cfe_evs_msg.h.

12.73.3.23 CFE_EVS_EnablePortsCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePortsCmd_t
```

Definition at line 1028 of file cfe_evs_msg.h.

12.73.3.24 CFE_EVS_HousekeepingTlm_Payload_t

```
typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t
```

Name Event Services Housekeeping Telemetry Packet

12.73.3.25 CFE_EVS_HousekeepingTlm_t

```
typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t
```

12.73.3.26 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](#)

12.73.3.27 CFE_EVS_LongEventTlm_Payload_t

```
typedef struct CFE_EVS_LongEventTlm_Payload CFE_EVS_LongEventTlm_Payload_t
```

Name Event Message Telemetry Packet (Long format)

12.73.3.28 CFE_EVS_LongEventTlm_t

```
typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t
```

12.73.3.29 CFE_EVS_NoArgsCmd_t

```
typedef struct CFE_EVS_NoArgsCmd CFE_EVS_NoArgsCmd_t
```

Command with no additional arguments.

12.73.3.30 CFE_EVS_NoopCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_NoopCmd_t
```

Definition at line 915 of file cfe_evs_msg.h.

12.73.3.31 CFE_EVS_PacketID_t

```
typedef struct CFE_EVS_PacketID CFE_EVS_PacketID_t
```

Telemetry packet structures

12.73.3.32 CFE_EVS_ResetAllFiltersCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFiltersCmd_t
```

Definition at line 1062 of file cfe_evs_msg.h.

12.73.3.33 CFE_EVS_ResetAppCounterCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounterCmd_t
```

Definition at line 1061 of file cfe_evs_msg.h.

12.73.3.34 CFE_EVS_ResetCountersCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCountersCmd_t
```

Definition at line 916 of file cfe_evs_msg.h.

12.73.3.35 CFE_EVS_ResetFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilterCmd_t
```

Definition at line 1090 of file cfe_evs_msg.h.

12.73.3.36 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](#)

12.73.3.37 CFE_EVS_SetEventFormatModeCmd_t

```
typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t
```

Set Event Format Mode Command.

12.73.3.38 CFE_EVS_SetFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilterCmd_t
```

Definition at line 1152 of file cfe_evs_msg.h.

12.73.3.39 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](#)

12.73.3.40 CFE_EVS_SetLogModeCmd_t

```
typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t
```

Set Log Mode Command.

12.73.3.41 CFE_EVS_ShortEventTlm_Payload_t

```
typedef struct CFE_EVS_ShortEventTlm_Payload CFE_EVS_ShortEventTlm_Payload_t
```

Name Event Message Telemetry Packet (Short format)

12.73.3.42 CFE_EVS_ShortEventTlm_t

```
typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t
```

12.73.3.43 CFE_EVS_WriteAppDataFileCmd_t

```
typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd_t
```

Write Event Services Application Information to File Command.

12.73.3.44 CFE_EVS_WriteLogFileCmd_t

```
typedef struct CFE_EVS_WriteLogFileCmd CFE_EVS_WriteLogFileCmd_t
```

Write Event Log to File Command.

12.74 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.

12.74.1 Detailed Description

Define CCSDS packet header types

- Avoid direct access for portability, use APIs
- Used to construct message structures

12.74.2 Typedef Documentation

12.74.2.1 CCSDS_ExtendedHeader_t

```
typedef struct CCSDS_ExtendedHeader CCSDS_ExtendedHeader_t
```

CCSDS packet extended header.

12.74.2.2 CCSDS_PrimaryHeader_t

```
typedef struct CCSDS_PrimaryHeader CCSDS_PrimaryHeader_t
```

CCSDS packet primary header.

12.75 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 }`
- 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_CD_BLOCKID_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) }`

12.75.1 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.

12.76 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.

12.76.1 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.

12.76.2 Macro Definition Documentation

12.76.2.1 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.

12.76.2.2 CFE_RESOURCEID_MAX

```
#define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK
```

Definition at line 65 of file cfe_resourceid_basevalue.h.

12.76.2.3 CFE_RESOURCEID_SHIFT

```
#define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT
```

Definition at line 64 of file cfe_resourceid_basevalue.h.

12.77 cfe/modules/sb/fsw/inc/cfe_sb_events.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
SB Subscribe API Max Destinations Exceeded Event ID.
- #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 Invalid Pipe Event ID.

- #define [CFE_SB_SETPPIPEOPTS_EID](#) 57
SB Set Pipe Opt API Not Owner Event ID.
- #define [CFE_SB_GETPIPEOPTS_ID_ERR_EID](#) 58
SB Set Pipe Opt API Success Event ID.
- #define [CFE_SB_GETPIPEOPTS_PTR_ERR_EID](#) 59
SB Get Pipe Opt API Invalid Pipe Event ID.
- #define [CFE_SB_GETPIPEOPTS_EID](#) 60
SB Get Pipe Opt API Invalid Pointer Event ID.
- #define [CFE_SB_GETPIPEOPTS_NAME_EID](#) 62
SB Get Pipe Opt API Success Event ID.
- #define [CFE_SB_GETPIPEOPTS_NULL_PTR_EID](#) 63
SB Get Pipe Opt API Invalid Pointer Event ID.
- #define [CFE_SB_GETPIPEOPTS_ID_ERR_EID](#) 64
SB Get Pipe Name API Invalid Pipe or Resource Event ID.
- #define [CFE_SB_GETPIPEIDBYNAME_EID](#) 65
SB Get Pipe ID By Name API Success Event ID.
- #define [CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID](#) 66
SB Get Pipe ID By Name API Invalid Pointer Event ID.
- #define [CFE_SB_GETPIPEIDBYNAME_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
SB Create Pipe API Name Taken Event ID.
- #define [CFE_SB_CR_PIPE_NO_FREE_EID](#) 70
SB Create Pipe API Queues Exhausted Event ID.

12.77.1 Detailed Description

cFE Software Bus Services Event IDs

12.77.2 Macro Definition Documentation

12.77.2.1 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_events.h.

12.77.2.2 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_events.h.

12.77.2.3 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_events.h.

12.77.2.4 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_events.h.

12.77.2.5 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_events.h`.

12.77.2.6 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_events.h`.

12.77.2.7 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_events.h`.

12.77.2.8 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_events.h.

12.77.2.9 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_events.h.

12.77.2.10 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_events.h.

12.77.2.11 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_events.h.

12.77.2.12 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_events.h.

12.77.2.13 CFE_SB_DSBL RTE1_EID

```
#define CFE_SB_DSBL RTE1_EID 36
```

SB Disable Route Command Invalid MsgId/PipId 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_events.h.

12.77.2.14 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_events.h.

12.77.2.15 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_events.h.

12.77.2.16 CFE_SB_DUP_SUBSCRIPT_EID

```
#define CFE_SB_DUP_SUBSCRIPT_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_events.h.

12.77.2.17 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_events.h.

12.77.2.18 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_events.h.

12.77.2.19 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_events.h.

12.77.2.20 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_events.h.

12.77.2.21 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_events.h.

12.77.2.22 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_events.h.

12.77.2.23 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_GetPipeldByName](#) success.

Definition at line 705 of file cfe_sb_events.h.

12.77.2.24 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_GetPipeldByName](#) failure due to name not found or ID mismatch. OVERLOADED

Definition at line 727 of file cfe_sb_events.h.

12.77.2.25 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_GetPipeldByName](#) failure due to invalid pointer.

Definition at line 716 of file cfe_sb_events.h.

12.77.2.26 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_events.h.

12.77.2.27 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_events.h.

12.77.2.28 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_events.h.

12.77.2.29 CFE_SB_GETPIPEOPTS_EID

```
#define CFE_SB_GETPIPEOPTS_EID 60
```

SB Get Pipe Opt API Success Event ID.

Type: DEBUG

Cause:

[CFE_SB_GetPipeOpts](#) success.

Definition at line 661 of file `cfe_sb_events.h`.

12.77.2.30 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_GetPipeOpts](#) failure due to invalid pipe ID.

Definition at line 639 of file `cfe_sb_events.h`.

12.77.2.31 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_GetPipeOpts](#) failure due to invalid pointer.

Definition at line 650 of file `cfe_sb_events.h`.

12.77.2.32 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_events.h.

12.77.2.33 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_events.h.

12.77.2.34 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_events.h.

12.77.2.35 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_events.h.

12.77.2.36 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_events.h.

12.77.2.37 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_events.h.

12.77.2.38 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_events.h.

12.77.2.39 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_events.h.

12.77.2.40 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_events.h.

12.77.2.41 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_events.h`.

12.77.2.42 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_events.h`.

12.77.2.43 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_events.h`.

12.77.2.44 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_events.h.

12.77.2.45 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_events.h.

12.77.2.46 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_events.h.

12.77.2.47 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_events.h.

12.77.2.48 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_events.h.

12.77.2.49 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_events.h.

12.77.2.50 CFE_SB_SETPPIPEOPTS_EID

```
#define CFE_SB_SETPPIPEOPTS_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_events.h.

12.77.2.51 CFE_SB_SETPPIPEOPTS_ID_ERR_EID

```
#define CFE_SB_SETPPIPEOPTS_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_events.h.

12.77.2.52 CFE_SB_SETPPIPEOPTS_OWNER_ERR_EID

```
#define CFE_SB_SETPPIPEOPTS_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_events.h.

12.77.2.53 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_events.h.

12.77.2.54 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_events.h.

12.77.2.55 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_events.h.

12.77.2.56 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_events.h.

12.77.2.57 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_events.h.

12.77.2.58 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_events.h.

12.77.2.59 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_events.h.

12.77.2.60 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_events.h.

12.77.2.61 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_events.h.

12.77.2.62 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_events.h.

12.77.2.63 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_events.h.

12.77.2.64 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_events.h.

12.77.2.65 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_events.h.

12.78 cfe/modules/sb/fsw/inc/cfe_sb_msg.h File Reference

```
#include "common_types.h"
#include "cfe_msg_hdr.h"
#include "cfe_sb_extern_typedefs.h"
#include "cfe_es_extern_typedefs.h"
```

Data Structures

- struct [CFE_SB_WriteFileInfoCmd_Payload](#)
Write File Info Command Payload.
- struct [CFE_SB_WriteFileInfoCmd](#)
Write File Info Command.
- struct [CFE_SB_RouteCmd_Payload](#)
Enable/Disable Route Command Payload.
- struct [CFE_SB_RouteCmd](#)
Enable/Disable Route Command.
- struct [CFE_SB_HousekeepingTlm_Payload](#)
- struct [CFE_SB_HousekeepingTlm](#)
- 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_StatsTlm](#)
- struct [CFE_SB_RoutingFileEntry](#)
SB Routing File Entry.
- struct [CFE_SB_MsgMapFileEntry](#)
SB Map File Entry.
- struct [CFE_SB_SingleSubscriptionTlm_Payload](#)
- struct [CFE_SB_SingleSubscriptionTlm](#)
- struct [CFE_SB_SubEntries](#)
SB Previous Subscriptions Entry.
- struct [CFE_SB_AllSubscriptionsTlm_Payload](#)
- struct [CFE_SB_AllSubscriptionsTlm](#)

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

Typedefs

- typedef CFE_MSG_CommandHeader_t CFE_SB_NoopCmd_t
- typedef CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd_t
- typedef CFE_MSG_CommandHeader_t CFE_SB_EnableSubReportingCmd_t
- typedef CFE_MSG_CommandHeader_t CFE_SB_DisableSubReportingCmd_t
- typedef CFE_MSG_CommandHeader_t CFE_SB_SendSbStatsCmd_t
- typedef CFE_MSG_CommandHeader_t CFE_SB_SendPrevSubsCmd_t
- typedef struct CFE_SB_WriteFileInfoCmd_Payload CFE_SB_WriteFileInfoCmd_Payload_t
Write File Info Command Payload.
- typedef struct CFE_SB_WriteFileInfoCmd CFE_SB_WriteFileInfoCmd_t
Write File Info Command.
- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteRoutingInfoCmd_t
- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WritePipeInfoCmd_t
- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteMapInfoCmd_t
- typedef struct CFE_SB_RouteCmd_Payload CFE_SB_RouteCmd_Payload_t
Enable/Disable Route Command Payload.
- typedef struct CFE_SB_RouteCmd CFE_SB_RouteCmd_t
Enable/Disable Route Command.
- typedef CFE_SB_RouteCmd_t CFE_SB_EnableRouteCmd_t
- typedef CFE_SB_RouteCmd_t CFE_SB_DisableRouteCmd_t
- typedef struct CFE_SB_HousekeepingTlm_Payload CFE_SB_HousekeepingTlm_Payload_t
- typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_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_StatsTlm CFE_SB_StatsTlm_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_SingleSubscriptionTlm CFE_SB_SingleSubscriptionTlm_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
- typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t

12.78.1 Detailed Description

Purpose: This header file contains structure definitions for all SB command and telemetry packets

Author: R.McGraw/SSI

12.78.2 Macro Definition Documentation

12.78.2.1 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_CM↔D_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 281 of file cfe_sb_msg.h.

12.78.2.2 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_SUB_CC](#)

Definition at line 438 of file cfe_sb_msg.h.

12.78.2.3 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 240 of file cfe_sb_msg.h.

12.78.2.4 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_SUBLIST_CC](#)

Definition at line 405 of file cfe_sb_msg.h.

12.78.2.5 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 76 of file cfe_sb_msg.h.

12.78.2.6 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 123 of file `cfe_sb_msg.h`.

12.78.2.7 CFE_SB_SEND_PREV_SUBS_CC

```
#define CFE_SB_SEND_PREV_SUBS_CC 11
```

Name Send Previous Subscriptions Command

This command generates a series of packets that contain information

regarding all subscriptions previously received by SB. This command is intended to be used only by the CFS 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 470 of file `cfe_sb_msg.h`.

12.78.2.8 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 157 of file cfe_sb_msg.h.

12.78.2.9 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 372 of file cfe_sb_msg.h.

12.78.2.10 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 326 of file cfe_sb_msg.h.

12.78.2.11 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 202 of file [cfe_sb_msg.h](#).

12.78.3 Ttypedef Documentation

12.78.3.1 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.

12.78.3.2 CFE_SB_AllSubscriptionsTlm_t

```
typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t
```

12.78.3.3 CFE_SB_DisableRouteCmd_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_DisableRouteCmd_t
```

Definition at line 545 of file cfe_sb_msg.h.

12.78.3.4 CFE_SB_DisableSubReportingCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_DisableSubReportingCmd_t
```

Definition at line 486 of file cfe_sb_msg.h.

12.78.3.5 CFE_SB_EnableRouteCmd_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_EnableRouteCmd_t
```

Definition at line 544 of file cfe_sb_msg.h.

12.78.3.6 CFE_SB_EnableSubReportingCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_EnableSubReportingCmd_t
```

Definition at line 485 of file cfe_sb_msg.h.

12.78.3.7 CFE_SB_HousekeepingTlm_Payload_t

```
typedef struct CFE_SB_HousekeepingTlm_Payload CFE_SB_HousekeepingTlm_Payload_t
```

Name Software Bus task housekeeping Packet

12.78.3.8 CFE_SB_HousekeepingTlm_t

```
typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t
```

12.78.3.9 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](#)

12.78.3.10 CFE_SB_NoopCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_NoopCmd_t
```

Definition at line 483 of file cfe_sb_msg.h.

12.78.3.11 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](#)

12.78.3.12 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.

12.78.3.13 CFE_SB_ResetCountersCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd_t
```

Definition at line 484 of file cfe_sb_msg.h.

12.78.3.14 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 Pipeld combination.

12.78.3.15 CFE_SB_RouteCmd_t

```
typedef struct CFE_SB_RouteCmd CFE_SB_RouteCmd_t
```

Enable/Disable Route Command.

12.78.3.16 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](#)

12.78.3.17 CFE_SB_SendPrevSubsCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_SendPrevSubsCmd_t
```

Definition at line 488 of file cfe_sb_msg.h.

12.78.3.18 CFE_SB_SendSbStatsCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_SendSbStatsCmd_t
```

Definition at line 487 of file cfe_sb_msg.h.

12.78.3.19 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](#)

12.78.3.20 CFE_SB_SingleSubscriptionTlm_t

```
typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscriptionTlm_t
```

12.78.3.21 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](#)

12.78.3.22 CFE_SB_StatsTlm_t

```
typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t
```

12.78.3.23 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](#)

12.78.3.24 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

12.78.3.25 CFE_SB_WriteFileInfoCmd_t

```
typedef struct CFE_SB_WriteFileInfoCmd CFE_SB_WriteFileInfoCmd_t
```

Write File Info Command.

12.78.3.26 CFE_SB_WriteMapInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteMapInfoCmd_t
```

Definition at line 514 of file cfe_sb_msg.h.

12.78.3.27 CFE_SB_WritePipeInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WritePipeInfoCmd_t
```

Definition at line 513 of file cfe_sb_msg.h.

12.78.3.28 CFE_SB_WriteRoutingInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteRoutingInfoCmd_t
```

Definition at line 512 of file cfe_sb_msg.h.

12.79 cfe/modules/tbl/fsw/inc/cfe_tbl_events.h File Reference

Macros

TBL event IDs

- #define `CFE_TBL_INIT_INF_EID` 1
TBL Initialization Event ID.
- #define `CFE_TBL_NOOP_INF_EID` 10
TBL No-op Command Success Event ID.
- #define `CFE_TBL_RESET_INF_EID` 11
TBL Reset Counters Command Success Event ID.
- #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

- #define `CFE_TBL_FILE_STD_HDR_ERR_EID` 54
TBL Load Table File Open Failure Event ID.
- #define `CFE_TBL_FILE_TBL_HDR_ERR_EID` 55
TBL Load Table File Read Standard Header Failure Event ID.
- #define `CFE_TBL_FAIL_HK_SEND_ERR_EID` 56
TBL Load Table File Read Table Header Failure Event ID.
- #define `CFE_TBL_NO SUCH_TABLE_ERR_EID` 57
TBL Send Housekeeping Command Transmit Failure 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_INTERNAL_ERROR_ERR_EID` 61
TBL Load Table Command Get Working Buffer Internal Failure 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
TBL Share Table Failed Event ID.
- #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.

12.79.1 Detailed Description

cFE Table Services Event IDs

12.79.2 Macro Definition Documentation

12.79.2.1 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 556 of file cfe_tbl_events.h.

12.79.2.2 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 462 of file cfe_tbl_events.h.

12.79.2.3 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_events.h.

12.79.2.4 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_events.h.

12.79.2.5 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 652 of file cfe_tbl_events.h.

12.79.2.6 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_events.h.

12.79.2.7 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 640 of file cfe_tbl_events.h.

12.79.2.8 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 664 of file cfe_tbl_events.h.

12.79.2.9 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 369 of file cfe_tbl_events.h.

12.79.2.10 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 544 of file cfe_tbl_events.h.

12.79.2.11 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_events.h.

12.79.2.12 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 686 of file cfe_tbl_events.h.

12.79.2.13 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_events.h.

12.79.2.14 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 474 of file cfe_tbl_events.h.

12.79.2.15 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_events.h.

12.79.2.16 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_events.h.

12.79.2.17 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_events.h.

12.79.2.18 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_events.h.

12.79.2.19 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 520 of file cfe_tbl_events.h.

12.79.2.20 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_events.h.

12.79.2.21 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 829 of file cfe_tbl_events.h.

12.79.2.22 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 580 of file cfe_tbl_events.h.

12.79.2.23 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 604 of file cfe_tbl_events.h.

12.79.2.24 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_events.h.

12.79.2.25 CFE_TBL_INTERNAL_ERROR_ERR_EID

```
#define CFE_TBL_INTERNAL_ERROR_ERR_EID 61
```

TBL Load Table Command Get Working Buffer Internal Failure Event ID.

Type: ERROR

Cause:

[TBL Load Table Command](#) failure due to internal get working buffer error.

Definition at line 358 of file cfe_tbl_events.h.

12.79.2.26 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_events.h.

12.79.2.27 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 450 of file cfe_tbl_events.h.

12.79.2.28 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_events.h.

12.79.2.29 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 486 of file cfe_tbl_events.h.

12.79.2.30 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 807 of file cfe_tbl_events.h.

12.79.2.31 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 796 of file cfe_tbl_events.h.

12.79.2.32 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_events.h.

12.79.2.33 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_events.h.

12.79.2.34 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 818 of file cfe_tbl_events.h.

12.79.2.35 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 741 of file cfe_tbl_events.h.

12.79.2.36 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 730 of file cfe_tbl_events.h.

12.79.2.37 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 567 of file cfe_tbl_events.h.

12.79.2.38 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 675 of file cfe_tbl_events.h.

12.79.2.39 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_events.h.

12.79.2.40 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 415 of file cfe_tbl_events.h.

12.79.2.41 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_events.h.

12.79.2.42 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_events.h.

12.79.2.43 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_events.h.

12.79.2.44 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 616 of file cfe_tbl_events.h.

12.79.2.45 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 628 of file cfe_tbl_events.h.

12.79.2.46 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_events.h.

12.79.2.47 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_events.h.

12.79.2.48 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 508 of file cfe_tbl_events.h.

12.79.2.49 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 785 of file cfe_tbl_events.h.

12.79.2.50 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 697 of file cfe_tbl_events.h.

12.79.2.51 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_events.h.

12.79.2.52 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 708 of file cfe_tbl_events.h.

12.79.2.53 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 774 of file cfe_tbl_events.h.

12.79.2.54 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_events.h.

12.79.2.55 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 532 of file cfe_tbl_events.h.

12.79.2.56 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 427 of file cfe_tbl_events.h.

12.79.2.57 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 719 of file cfe_tbl_events.h.

12.79.2.58 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 592 of file cfe_tbl_events.h.

12.79.2.59 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 752 of file cfe_tbl_events.h.

12.79.2.60 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_events.h.

12.79.2.61 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_events.h.

12.79.2.62 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 763 of file cfe_tbl_events.h.

12.79.2.63 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_events.h.

12.79.2.64 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 380 of file cfe_tbl_events.h.

12.79.2.65 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_events.h.

12.79.2.66 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_events.h.

12.79.2.67 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 391 of file cfe_tbl_events.h.

12.79.2.68 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 402 of file cfe_tbl_events.h.

12.79.2.69 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 438 of file cfe_tbl_events.h.

12.79.2.70 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 497 of file cfe_tbl_events.h.

12.80 cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h File Reference

```
#include "common_types.h"
#include "cfe_msg_hdr.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_time_extern_typedefs.h"
```

Data Structures

- struct [CFE_TBL_NoArgsCmd](#)
Generic "no arguments" command.
- struct [CFE_TBL_LoadCmd_Payload](#)
Load Table Command Payload.
- struct [CFE_TBL_LoadCmd](#)
Load Table Command.
- struct [CFE_TBL_DumpCmd_Payload](#)
Dump Table Command Payload.
- struct [CFE_TBL_DumpCmd](#)
- struct [CFE_TBL_ValidateCmd_Payload](#)
Validate Table Command Payload.
- struct [CFE_TBL_ValidateCmd](#)
Validate Table Command.
- struct [CFE_TBL_ActivateCmd_Payload](#)
Activate Table Command Payload.
- struct [CFE_TBL_ActivateCmd](#)
Activate Table Command.
- struct [CFE_TBL_DumpRegistryCmd_Payload](#)
Dump Registry Command Payload.

- struct [CFE_TBL_DumpRegistryCmd](#)
Dump Registry Command.
- struct [CFE_TBL_SendRegistryCmd_Payload](#)
Send Table Registry Command Payload.
- struct [CFE_TBL_SendRegistryCmd](#)
Send Table Registry Command.
- struct [CFE_TBL_DeleteCDSCmd_Payload](#)
Delete Critical Table CDS Command Payload.
- struct [CFE_TBL_DeleteCDSCmd](#)
Delete Critical Table CDS Command.
- struct [CFE_TBL_AbortLoadCmd_Payload](#)
Abort Load Command Payload.
- struct [CFE_TBL_AbortLoadCmd](#)
Abort Load Command.
- struct [CFE_TBL_NotifyCmd_Payload](#)
Table Management Notification Command Payload.
- struct [CFE_TBL_NotifyCmd](#)
- struct [CFE_TBL_HousekeepingTlm_Payload](#)
- struct [CFE_TBL_HousekeepingTlm](#)
- struct [CFE_TBL_TblRegPacket_Payload](#)
- struct [CFE_TBL_TableRegistryTlm](#)

Macros

Table Services Command Codes

- #define [CFE_TBL_NOOP_CC](#) 0
- #define [CFE_TBL_RESET_COUNTERS_CC](#) 1
- #define [CFE_TBL_LOAD_CC](#) 2
- #define [CFE_TBL_DUMP_CC](#) 3
- #define [CFE_TBL_VALIDATE_CC](#) 4
- #define [CFE_TBL_ACTIVATE_CC](#) 5
- #define [CFE_TBL_DUMP_REGISTRY_CC](#) 6
- #define [CFE_TBL_SEND_REGISTRY_CC](#) 7
- #define [CFE_TBL_DELETE_CDS_CC](#) 8
- #define [CFE_TBL_ABORT_LOAD_CC](#) 9

Typedefs

- typedef struct [CFE_TBL_NoArgsCmd](#) [CFE_TBL_NoArgsCmd_t](#)
Generic "no arguments" command.
- typedef [CFE_TBL_NoArgsCmd_t](#) [CFE_TBL_NoopCmd_t](#)
- typedef [CFE_TBL_NoArgsCmd_t](#) [CFE_TBL_ResetCountersCmd_t](#)
- typedef struct [CFE_TBL_LoadCmd_Payload](#) [CFE_TBL_LoadCmd_Payload_t](#)
Load Table Command Payload.
- typedef struct [CFE_TBL_LoadCmd](#) [CFE_TBL_LoadCmd_t](#)
Load Table Command.
- typedef struct [CFE_TBL_DumpCmd_Payload](#) [CFE_TBL_DumpCmd_Payload_t](#)
Dump Table Command Payload.

- typedef struct `CFE_TBL_DumpCmd` `CFE_TBL_DumpCmd_t`
- typedef struct `CFE_TBL_ValidateCmd_Payload` `CFE_TBL_ValidateCmd_Payload_t`
Validate Table Command Payload.
- typedef struct `CFE_TBL_ValidateCmd` `CFE_TBL_ValidateCmd_t`
Validate Table Command.
- typedef struct `CFE_TBL_ActivateCmd_Payload` `CFE_TBL_ActivateCmd_Payload_t`
Activate Table Command Payload.
- typedef struct `CFE_TBL_ActivateCmd` `CFE_TBL_ActivateCmd_t`
Activate Table Command.
- typedef struct `CFE_TBL_DumpRegistryCmd_Payload` `CFE_TBL_DumpRegistryCmd_Payload_t`
Dump Registry Command Payload.
- typedef struct `CFE_TBL_DumpRegistryCmd` `CFE_TBL_DumpRegistryCmd_t`
Dump Registry Command.
- typedef struct `CFE_TBL_SendRegistryCmd_Payload` `CFE_TBL_SendRegistryCmd_Payload_t`
Send Table Registry Command Payload.
- typedef struct `CFE_TBL_SendRegistryCmd` `CFE_TBL_SendRegistryCmd_t`
Send Table Registry Command.
- typedef struct `CFE_TBL_DeleteCDSCmd_Payload` `CFE_TBL_DeleteCDSCmd_Payload_t`
Delete Critical Table CDS Command Payload.
- typedef struct `CFE_TBL_DeleteCDSCmd` `CFE_TBL_DeleteCDSCmd_t`
Delete Critical Table CDS Command.
- typedef struct `CFE_TBL_AbortLoadCmd_Payload` `CFE_TBL_AbortLoadCmd_Payload_t`
Abort Load Command Payload.
- typedef struct `CFE_TBL_AbortLoadCmd` `CFE_TBL_AbortLoadCmd_t`
Abort Load Command.
- typedef struct `CFE_TBL_NotifyCmd_Payload` `CFE_TBL_NotifyCmd_Payload_t`
Table Management Notification Command Payload.
- typedef struct `CFE_TBL_NotifyCmd` `CFE_TBL_NotifyCmd_t`
- typedef struct `CFE_TBL_HousekeepingTlm_Payload` `CFE_TBL_HousekeepingTlm_Payload_t`
- typedef struct `CFE_TBL_HousekeepingTlm` `CFE_TBL_HousekeepingTlm_t`
- typedef struct `CFE_TBL_TblRegPacket_Payload` `CFE_TBL_TblRegPacket_Payload_t`
- typedef struct `CFE_TBL_TableRegistryTlm` `CFE_TBL_TableRegistryTlm_t`

12.80.1 Detailed Description

Purpose: cFE Table Services (TBL) SB message definitions header file

Author: D.Kobe/Hammers

Notes:

12.80.2 Macro Definition Documentation

12.80.2.1 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 473 of file cfe_tbl_msg.h.

12.80.2.2 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 311 of file `cfe_tbl_msg.h`.

12.80.2.3 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 434 of file `cfe_tbl_msg.h`.

12.80.2.4 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 214 of file `cfe_tbl_msg.h`.

12.80.2.5 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 355 of file `cfe_tbl_msg.h`.

12.80.2.6 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 171 of file `cfe_tbl_msg.h`.

12.80.2.7 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 80 of file [cfe_tbl_msg.h](#).

12.80.2.8 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_ValCompltdCtr)

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 121 of file `cfe_tbl_msg.h`.

12.80.2.9 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 390 of file cfe_tbl_msg.h.

12.80.2.10 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 271 of file `cfe_tbl_msg.h`.

12.80.3 Typedef Documentation

12.80.3.1 CFE_TBL_AbortLoadCmd_Payload_t

```
typedef struct CFE_TBL_AbortLoadCmd_Payload CFE_TBL_AbortLoadCmd_Payload_t
```

Abort Load Command Payload.

For command details, see [CFE_TBL_ABORT_LOAD_CC](#)

12.80.3.2 CFE_TBL_AbortLoadCmd_t

```
typedef struct CFE_TBL_AbortLoadCmd CFE_TBL_AbortLoadCmd_t
```

Abort Load Command.

12.80.3.3 CFE_TBL_ActivateCmd_Payload_t

```
typedef struct CFE_TBL_ActivateCmd_Payload CFE_TBL_ActivateCmd_Payload_t
```

Activate Table Command Payload.

For command details, see [CFE_TBL_ACTIVATE_CC](#)

12.80.3.4 CFE_TBL_ActivateCmd_t

```
typedef struct CFE_TBL_ActivateCmd CFE_TBL_ActivateCmd_t
```

Activate Table Command.

12.80.3.5 CFE_TBL_DelCDSCmd_Payload_t

```
typedef struct CFE_TBL_DelCDSCmd_Payload CFE_TBL_DelCDSCmd_Payload_t
```

Delete Critical Table CDS Command Payload.

For command details, see [CFE_TBL_DELETE_CDS_CC](#)

12.80.3.6 CFE_TBL_DeleteCDSCmd_t

```
typedef struct CFE_TBL_DeleteCDSCmd CFE_TBL_DeleteCDSCmd_t
```

Delete Critical Table CDS Command.

12.80.3.7 CFE_TBL_DumpCmd_Payload_t

```
typedef struct CFE_TBL_DumpCmd_Payload CFE_TBL_DumpCmd_Payload_t
```

Dump Table Command Payload.

For command details, see [CFE_TBL_DUMP_CC](#)

12.80.3.8 CFE_TBL_DumpCmd_t

```
typedef struct CFE_TBL_DumpCmd CFE_TBL_DumpCmd_t
```

/brief Dump Table Command

12.80.3.9 CFE_TBL_DumpRegistryCmd_Payload_t

```
typedef struct CFE_TBL_DumpRegistryCmd_Payload CFE_TBL_DumpRegistryCmd_Payload_t
```

Dump Registry Command Payload.

For command details, see [CFE_TBL_DUMP_REGISTRY_CC](#)

12.80.3.10 CFE_TBL_DumpRegistryCmd_t

```
typedef struct CFE_TBL_DumpRegistryCmd CFE_TBL_DumpRegistryCmd_t
```

Dump Registry Command.

12.80.3.11 CFE_TBL_HousekeepingTlm_Payload_t

```
typedef struct CFE_TBL_HousekeepingTlm_Payload CFE_TBL_HousekeepingTlm_Payload_t
```

Name Table Services Housekeeping Packet

12.80.3.12 CFE_TBL_HousekeepingTlm_t

```
typedef struct CFE_TBL_HousekeepingTlm CFE_TBL_HousekeepingTlm_t
```

12.80.3.13 CFE_TBL_LoadCmd_Payload_t

```
typedef struct CFE_TBL_LoadCmd_Payload CFE_TBL_LoadCmd_Payload_t
```

Load Table Command Payload.

For command details, see [CFE_TBL_LOAD_CC](#)

12.80.3.14 CFE_TBL_LoadCmd_t

```
typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t
```

Load Table Command.

12.80.3.15 CFE_TBL_NoArgsCmd_t

```
typedef struct CFE_TBL_NoArgsCmd CFE_TBL_NoArgsCmd_t
```

Generic "no arguments" command.

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see [CFE_TBL_NOOP_CC](#))
3. The Reset Counters Command (For details, see [CFE_TBL_RESET_COUNTERS_CC](#))

12.80.3.16 CFE_TBL_NoopCmd_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_NoopCmd_t
```

Definition at line 500 of file cfe_tbl_msg.h.

12.80.3.17 CFE_TBL_NotifyCmd_Payload_t

```
typedef struct CFE_TBL_NotifyCmd_Payload CFE_TBL_NotifyCmd_Payload_t
```

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).

12.80.3.18 CFE_TBL_NotifyCmd_t

```
typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t
```

/brief Table Management Notification Command

12.80.3.19 CFE_TBL_ResetCountersCmd_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_ResetCountersCmd_t
```

Definition at line 501 of file cfe_tbl_msg.h.

12.80.3.20 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](#)

12.80.3.21 CFE_TBL_SendRegistryCmd_t

```
typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t
```

Send Table Registry Command.

12.80.3.22 CFE_TBL_TableRegistryTlm_t

```
typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t
```

12.80.3.23 CFE_TBL_TblRegPacket_Payload_t

```
typedef struct CFE_TBL_TblRegPacket_Payload CFE_TBL_TblRegPacket_Payload_t
```

Name Table Registry Info Packet

12.80.3.24 CFE_TBL_ValidateCmd_Payload_t

```
typedef struct CFE_TBL_ValidateCmd_Payload CFE_TBL_ValidateCmd_Payload_t
```

Validate Table Command Payload.

For command details, see [CFE_TBL_VALIDATE_CC](#)

12.80.3.25 CFE_TBL_ValidateCmd_t

```
typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t
```

Validate Table Command.

12.81 cfe/modules/time/fsw/inc/cfe_time_events.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_1HZ_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_1HZ_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.

12.81.1 Detailed Description

cFE Time Services Event IDs

12.81.2 Macro Definition Documentation

12.81.2.1 CFE_TIME_1HZ_CFG_EID

```
#define CFE_TIME_1HZ_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_events.h.

12.81.2.2 CFE_TIME_1HZ_EID

```
#define CFE_TIME_1HZ_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_events.h.

12.81.2.3 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_events.h](#).

12.81.2.4 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_events.h](#).

12.81.2.5 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_events.h](#).

12.81.2.6 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_events.h.

12.81.2.7 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_events.h.

12.81.2.8 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_events.h.

12.81.2.9 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_events.h`.

12.81.2.10 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_events.h`.

12.81.2.11 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_events.h`.

12.81.2.12 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_events.h.

12.81.2.13 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_events.h.

12.81.2.14 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_events.h.

12.81.2.15 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_events.h.

12.81.2.16 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_events.h.

12.81.2.17 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_events.h.

12.81.2.18 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_events.h.

12.81.2.19 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_events.h.

12.81.2.20 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_events.h.

12.81.2.21 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_events.h.

12.81.2.22 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_events.h.

12.81.2.23 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_events.h.

12.81.2.24 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_events.h.

12.81.2.25 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_events.h.

12.81.2.26 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_events.h.

12.81.2.27 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_events.h.

12.81.2.28 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_events.h.

12.81.2.29 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_events.h.

12.81.2.30 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_events.h.

12.81.2.31 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_events.h.

12.81.2.32 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_events.h.

12.81.2.33 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_events.h.

12.81.2.34 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_events.h.

12.81.2.35 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_events.h.

12.81.2.36 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_events.h.

12.82 cfe/modules/time/fsw/inc/cfe_time_msg.h File Reference

```
#include "common_types.h"
#include "cfe-msg-hdr.h"
#include "cfe_time_extern_typedefs.h"
```

Data Structures

- struct [CFE_TIME_NoArgsCmd](#)
Generic no argument command.
- struct [CFE_TIME_LeapsCmd_Payload](#)
Set leap seconds command payload.
- struct [CFE_TIME_SetLeapSecondsCmd](#)
Set leap seconds command.
- struct [CFE_TIME_StateCmd_Payload](#)
Set clock state command payload.
- struct [CFE_TIME_SetStateCmd](#)
Set clock state command.
- struct [CFE_TIME_SourceCmd_Payload](#)
Set time data source command payload.
- struct [CFE_TIME_SetSourceCmd](#)
Set time data source command.
- struct [CFE_TIME_SignalCmd_Payload](#)
Set tone signal source command payload.
- struct [CFE_TIME_SetSignalCmd](#)
Set tone signal source command.
- struct [CFE_TIME_TimeCmd_Payload](#)
Generic seconds, microseconds command payload.

- struct **CFE_TIME_TimeCmd**
Generic seconds, microseconds argument command.
- struct **CFE_TIME_OneHzAdjustmentCmd_Payload**
Generic seconds, subseconds command payload.
- struct **CFE_TIME_OneHzAdjustmentCmd**
Generic seconds, subseconds adjustment command.
- struct **CFE_TIME_ToneDataCmd_Payload**
Time at tone data command payload.
- struct **CFE_TIME_ToneDataCmd**
Time at tone data command.
- struct **CFE_TIME_HousekeepingTlm_Payload**
- struct **CFE_TIME_HousekeepingTlm**
- struct **CFE_TIME_DiagnosticTlm_Payload**
- struct **CFE_TIME_DiagnosticTlm**

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.

Time Services Command Codes

- #define **CFE_TIME_NOOP_CC** 0 /* no-op command */

- #define CFE_TIME_RESET_COUNTERS_CC 1 /* reset counters */
- #define CFE_TIME_SEND_DIAGNOSTIC_TLM_CC 2 /* request diagnostic hk telemetry */
- #define CFE_TIME_SET_SOURCE_CC 3 /* set clock source (int vs ext) */
- #define CFE_TIME_SET_STATE_CC 4 /* set clock state */
- #define CFE_TIME_ADD_DELAY_CC 5 /* add tone delay value */
- #define CFE_TIME_SUB_DELAY_CC 6 /* sub tone delay value */
- #define CFE_TIME_SET_TIME_CC 7 /* set time */
- #define CFE_TIME_SET_MET_CC 8 /* set MET */
- #define CFE_TIME_SET_STCF_CC 9 /* set STCF */
- #define CFE_TIME_SET_LEAP_SECONDS_CC 10 /* set Leap Seconds */
- #define CFE_TIME_ADD_ADJUST_CC 11 /* add one time STCF adjustment */
- #define CFE_TIME_SUB_ADJUST_CC 12 /* subtract one time STCF adjustment */
- #define CFE_TIME_ADD_1HZ_ADJUSTMENT_CC 13 /* add 1Hz STCF adjustment */
- #define CFE_TIME_SUB_1HZ_ADJUSTMENT_CC 14 /* subtract 1Hz STCF adjustment */
- #define CFE_TIME_SET_SIGNAL_CC 15 /* set clock signal (pri vs red) */

Typedefs

- typedef struct CFE_TIME_NoArgsCmd CFE_TIME_NoArgsCmd_t

Generic no argument command.
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_NoopCmd_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCountersCmd_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticCmd_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_1HzCmd_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ToneSignalCmd_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_FakeToneCmd_t
- typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t

Set leap seconds command payload.
- typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t

Set leap seconds command.
- typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t

Set clock state command payload.
- typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t

Set clock state command.
- typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t

Set time data source command payload.
- typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t

Set time data source command.
- typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t

Set tone signal source command payload.
- typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t

Set tone signal source command.
- typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t

Generic seconds, microseconds command payload.
- typedef struct CFE_TIME_TimeCmd CFE_TIME_TimeCmd_t

Generic seconds, microseconds argument command.
- typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelayCmd_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelayCmd_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMETCmd_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCFCmd_t

- `typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjustCmd_t`
- `typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjustCmd_t`
- `typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTimeCmd_t`
- `typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t`
Generic seconds, subseconds command payload.
- `typedef struct CFE_TIME_OneHzAdjustmentCmd CFE_TIME_OneHzAdjustmentCmd_t`
Generic seconds, subseconds adjustment command.
- `typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustmentCmd_t`
- `typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustmentCmd_t`
- `typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t`
Time at tone data command payload.
- `typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t`
Time at tone data command.
- `typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t`
- `typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t`
- `typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t`
- `typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t`

12.82.1 Detailed Description

Purpose: cFE Time Services (TIME) SB message definitions header file

Author: S.Walling/Microtel

Notes:

12.82.2 Macro Definition Documentation

12.82.2.1 CFE_TIME_ADD_1HZ_ADJUSTMENT_CC

```
#define CFE_TIME_ADD_1HZ_ADJUSTMENT_CC 13 /* add 1Hz STCF adjustment */
```

Name Add Delta to Spacecraft Time Correlation Factor each 1Hz

Description

This command has been updated to take actual sub-seconds ($1/2^{32}$ seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by adding the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

Command Mnemonic(s) \$sc_\$cpu_TIME_Add1HzSTCF

Command Structure

[CFE_TIME_Add1HZAdjustmentCmd_t](#)

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_TIME_CMDPC** - command execution counter will increment
- **\$sc_\$cpu_TIME_STCFSecs** - Housekeeping Telemetry point indicating new STCF seconds value
- **\$sc_\$cpu_TIME_STCFSubsecs** - Housekeeping Telemetry point indicating new STCF subseconds value
- The [CFE_TIME_1HZ_EID](#) informational event message will be generated

Error Conditions

- 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_1HZ_CFG_EID](#))

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

[CFE_TIME_ADD_ADJUST_CC](#), [CFE_TIME_SUB_ADJUST_CC](#), [CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#)

Definition at line 612 of file cfe_time_msg.h.

12.82.2.2 CFE_TIME_ADD_ADJUST_CC

```
#define CFE_TIME_ADD_ADJUST_CC 11 /* add one time STCF adjustment */
```

Name Add Delta to Spacecraft Time Correlation Factor

Description

This command adjusts the Spacecraft Time Correlation Factor (STCF) by adding the specified value. The new STCF takes effect immediately upon execution of this command.

Command Mnemonic(s) \$sc_\$cpu_TIME_AddSTCFAdj

Command Structure

[CFE_TIME_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_1HZ_ADJUSTMENT_CC](#),
[CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#)

Definition at line 532 of file `cfe_time_msg.h`.

12.82.2.3 CFE_TIME_ADD_DELAY_CC

```
#define CFE_TIME_ADD_DELAY_CC 5 /* add tone delay value */
```

Name Add Time to Tone Time Delay

Description

This command is used to factor out a known, predictable latency between the Time Server and a particular Time Client. The correction is applied (added) to the current time calculation for Time Clients, so this command has no meaning for Time Servers. Each Time Client can have a unique latency setting. The latency value is a positive number of seconds and microseconds that represent the deviation from the time maintained by the Time Server.

Command Mnemonic(s) \$sc_\$cpu_TIME_AddClockLat**Command Structure**[CFE_TIME_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_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_SUB_DELAY_CC](#)

Definition at line 301 of file cfe_time_msg.h.

12.82.2.4 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 77 of file `cfe_time_msg.h`.

12.82.2.5 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 122 of file cfe_time_msg.h.

12.82.2.6 CFE_TIME_SEND_DIAGNOSTIC_TLM_CC

```
#define CFE_TIME_SEND_DIAGNOSTIC_TLM_CC 2 /* request diagnostic hk telemetry */
```

Name Request TIME Diagnostic Telemetry

Description

This command requests that the Time Service generate a message containing various data values not included in the normal Time Service housekeeping message. The command requests only a single copy of the diagnostic message. Refer to [CFE_TIME_DiagnosticTlm_t](#) for a description of the Time Service diagnostic message contents.

Command Mnemonic(s) \$sc_\$cpu_TIME_RequestDiag

Command Structure

[CFE_TIME_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 156 of file `cfe_time_msg.h`.

12.82.2.7 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 496 of file `cfe_time_msg.h`.

12.82.2.8 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 424 of file `cfe_time_msg.h`.

12.82.2.9 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 702 of file cfe_time_msg.h.

12.82.2.10 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 206 of file `cfe_time_msg.h`.

12.82.2.11 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 263 of file `cfe_time_msg.h`.

12.82.2.12 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 461 of file `cfe_time_msg.h`.

12.82.2.13 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 sub-seconds value
- The [CFE_TIME_TIME_EID](#) informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- 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 384 of file `cfe_time_msg.h`.

12.82.2.14 CFE_TIME_SUB_1HZ_ADJUSTMENT_CC

```
#define CFE_TIME_SUB_1HZ_ADJUSTMENT_CC 14 /* subtract 1Hz STCF adjustment */
```

Name Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

Description

This command has been updated to take actual sub-seconds ($1/2^{32}$ seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by subtracting the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

Command Mnemonic(s) \$sc_\$cpu_TIME_Sub1HzSTCF**Command Structure**[CFE_TIME_Sub1HZAdjustmentCmd_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry: Successful execution of this command may be verified with the following telemetry:

- **\$sc_\$cpu_TIME_CMDPC** - command execution counter will increment
- **\$sc_\$cpu_TIME_STCFSecs** - Housekeeping Telemetry point indicating new STCF seconds value
- **\$sc_\$cpu_TIME_STCFSubsecs** - Housekeeping Telemetry point indicating new STCF subseconds value
- The [CFE_TIME_1HZ_EID](#) informational event message will be generated

Error Conditions

- 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_1HZ_CFG_EID](#))

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also[CFE_TIME_ADD_ADJUST_CC](#), [CFE_TIME_SUB_ADJUST_CC](#), [CFE_TIME_ADD_1HZ_ADJUSTMENT_CC](#)

Definition at line 660 of file cfe_time_msg.h.

12.82.2.15 CFE_TIME_SUB_ADJUST_CC

```
#define CFE_TIME_SUB_ADJUST_CC 12 /* subtract one time STCF adjustment */
```

Name Subtract Delta from Spacecraft Time Correlation Factor

Description

This command adjusts the Spacecraft Time Correlation Factor (STCF) by subtracting the specified value. The new STCF takes effect immediately upon execution of this command.

Command Mnemonic(s) \$sc_\$cpu_TIME_SubSTCFAdj

Command Structure

[CFE_TIME_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_1HZ_ADJUSTMENT_CC](#), [CFE_TIME_SUB_1HZ_ADJUSTMENT_CC](#)

Definition at line 566 of file `cfe_time_msg.h`.

12.82.2.16 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_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_ADD_DELAY_CC](#)

Definition at line 339 of file cfe_time_msg.h.

12.82.3 Typedef Documentation

12.82.3.1 CFE_TIME_1HzCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_1HzCmd_t
```

Definition at line 743 of file cfe_time_msg.h.

12.82.3.2 CFE_TIME_Add1HZAdjustmentCmd_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustmentCmd_t
```

Definition at line 878 of file cfe_time_msg.h.

12.82.3.3 CFE_TIME_AddAdjustCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjustCmd_t
```

Definition at line 850 of file cfe_time_msg.h.

12.82.3.4 CFE_TIME_AddDelayCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelayCmd_t
```

Definition at line 846 of file cfe_time_msg.h.

12.82.3.5 CFE_TIME_DiagnosticTlm_Payload_t

```
typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t
```

Name Time Services Diagnostics Packet

12.82.3.6 CFE_TIME_DiagnosticTlm_t

```
typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t
```

12.82.3.7 CFE_TIME_FakeToneCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_FakeToneCmd_t
```

Definition at line 745 of file cfe_time_msg.h.

12.82.3.8 CFE_TIME_HousekeepingTlm_Payload_t

```
typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t
```

Name Time Services Housekeeping Packet

12.82.3.9 CFE_TIME_HousekeepingTlm_t

```
typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t
```

12.82.3.10 CFE_TIME_LeapsCmd_Payload_t

```
typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t
```

Set leap seconds command payload.

12.82.3.11 CFE_TIME_NoArgsCmd_t

```
typedef struct CFE_TIME_NoArgsCmd CFE_TIME_NoArgsCmd_t
```

Generic no argument command.

12.82.3.12 CFE_TIME_NoopCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_NoopCmd_t
```

Definition at line 740 of file cfe_time_msg.h.

12.82.3.13 CFE_TIME_OneHzAdjustmentCmd_Payload_t

```
typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t
```

Generic seconds, subseconds command payload.

12.82.3.14 CFE_TIME_OneHzAdjustmentCmd_t

```
typedef struct CFE_TIME_OneHzAdjustmentCmd CFE_TIME_OneHzAdjustmentCmd_t
```

Generic seconds, subseconds adjustment command.

12.82.3.15 CFE_TIME_ResetCountersCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCountersCmd_t
```

Definition at line 741 of file cfe_time_msg.h.

12.82.3.16 CFE_TIME_SendDiagnosticCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticCmd_t
```

Definition at line 742 of file cfe_time_msg.h.

12.82.3.17 CFE_TIME_SetLeapSecondsCmd_t

```
typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t
```

Set leap seconds command.

12.82.3.18 CFE_TIME_SetMETCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMETCmd_t
```

Definition at line 848 of file cfe_time_msg.h.

12.82.3.19 CFE_TIME_SetSignalCmd_t

```
typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t
```

Set tone signal source command.

12.82.3.20 CFE_TIME_SetSourceCmd_t

```
typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t
```

Set time data source command.

12.82.3.21 CFE_TIME_SetStateCmd_t

```
typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t
```

Set clock state command.

12.82.3.22 CFE_TIME_SetSTCFCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCFCmd_t
```

Definition at line 849 of file cfe_time_msg.h.

12.82.3.23 CFE_TIME_SetTimeCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTimeCmd_t
```

Definition at line 852 of file cfe_time_msg.h.

12.82.3.24 CFE_TIME_SignalCmd_Payload_t

```
typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t
```

Set tone signal source command payload.

12.82.3.25 CFE_TIME_SourceCmd_Payload_t

```
typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t
```

Set time data source command payload.

12.82.3.26 CFE_TIME_StateCmd_Payload_t

```
typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t
```

Set clock state command payload.

12.82.3.27 CFE_TIME_Sub1HZAdjustmentCmd_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustmentCmd_t
```

Definition at line 879 of file cfe_time_msg.h.

12.82.3.28 CFE_TIME_SubAdjustCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjustCmd_t
```

Definition at line 851 of file cfe_time_msg.h.

12.82.3.29 CFE_TIME_SubDelayCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelayCmd_t
```

Definition at line 847 of file cfe_time_msg.h.

12.82.3.30 CFE_TIME_TimeCmd_Payload_t

```
typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t
```

Generic seconds, microseconds command payload.

12.82.3.31 CFE_TIME_TimeCmd_t

```
typedef struct CFE_TIME_TimeCmd CFE_TIME_TimeCmd_t
```

Generic seconds, microseconds argument command.

12.82.3.32 CFE_TIME_ToneDataCmd_Payload_t

```
typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t
```

Time at tone data command payload.

12.82.3.33 CFE_TIME_ToneDataCmd_t

```
typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t
```

Time at tone data command.

12.82.3.34 CFE_TIME_ToneSignalCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ToneSignalCmd_t
```

Definition at line 744 of file cfe_time_msg.h.

12.83 osal/docs/src/osal_frontpage.dox File Reference**12.84 osal/docs/src/osal_fs.dox File Reference****12.85 osal/docs/src/osal_timer.dox File Reference****12.86 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)`

12.86.1 Detailed Description

Purpose: Unit specification for common types.

Design Notes: Assumes make file has defined processor family

12.86.2 Macro Definition Documentation

12.86.2.1 _EXTENSION_

```
#define _EXTENSION_
```

Definition at line 65 of file common_types.h.

12.86.2.2 CompileTimeAssert

```
#define CompileTimeAssert(  
    Condition,  
    Message ) typedef char Message[(Condition) ? 1 : -1]
```

Definition at line 48 of file common_types.h.

12.86.2.3 OS_PRINTF

```
#define OS_PRINTF(  
    n,  
    m )
```

Definition at line 67 of file common_types.h.

12.86.2.4 OS_USED

```
#define OS_USED
```

Definition at line 66 of file common_types.h.

12.86.2.5 OSAL_BLOCKCOUNT_C

```
#define OSAL_BLOCKCOUNT_C(  
    X ) ((osal_blockcount_t)(X))
```

Definition at line 172 of file common_types.h.

12.86.2.6 OSAL_INDEX_C

```
#define OSAL_INDEX_C(  
    X ) ((osal_index_t)(X))
```

Definition at line 173 of file common_types.h.

12.86.2.7 OSAL_OBJTYPE_C

```
#define OSAL_OBJTYPE_C(  
    X ) ((osal_objtype_t)(X))
```

Definition at line 174 of file common_types.h.

12.86.2.8 OSAL_SIZE_C

```
#define OSAL_SIZE_C(  
    X ) ((size_t)(X))
```

Definition at line 171 of file common_types.h.

12.86.2.9 OSAL_STATUS_C

```
#define OSAL_STATUS_C(  
    X ) ((osal_status_t)(X))
```

Definition at line 175 of file common_types.h.

12.86.3 Typedef Documentation

12.86.3.1 cpuaddr

```
typedef uintptr_t cpuaddr
```

Definition at line 88 of file common_types.h.

12.86.3.2 cpudiff

```
typedef ptrdiff_t cpudiff
```

Definition at line 90 of file common_types.h.

12.86.3.3 cpusize

```
typedef size_t cpusize
```

Definition at line 89 of file common_types.h.

12.86.3.4 int16

```
typedef int16_t int16
```

Definition at line 80 of file common_types.h.

12.86.3.5 int32

```
typedef int32_t int32
```

Definition at line 81 of file common_types.h.

12.86.3.6 int64

```
typedef int64_t int64
```

Definition at line 82 of file common_types.h.

12.86.3.7 int8

```
typedef int8_t int8
```

Definition at line 79 of file common_types.h.

12.86.3.8 intptr

```
typedef intptr_t intptr
```

Definition at line 87 of file common_types.h.

12.86.3.9 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.

12.86.3.10 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.

12.86.3.11 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.

12.86.3.12 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_ObjectIdToIndex\(\)](#).

Definition at line 126 of file common_types.h.

12.86.3.13 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.

12.86.3.14 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.

12.86.3.15 uint16

```
typedef uint16_t uint16
```

Definition at line 84 of file common_types.h.

12.86.3.16 uint32

```
typedef uint32_t uint32
```

Definition at line 85 of file common_types.h.

12.86.3.17 uint64

```
typedef uint64_t uint64
```

Definition at line 86 of file common_types.h.

12.86.3.18 uint8

```
typedef uint8_t uint8
```

Definition at line 83 of file common_types.h.

12.86.4 Function Documentation**12.86.4.1 CompileTimeAssert() [1/9]**

```
CompileTimeAssert (
    sizeof(uint8) == 1,
    TypeUint8WrongSize )
```

12.86.4.2 CompileTimeAssert() [2/9]

```
CompileTimeAssert (
    sizeof(uint16) == 2,
    TypeUint16WrongSize )
```

12.86.4.3 CompileTimeAssert() [3/9]

```
CompileTimeAssert (
    sizeof(uint32) == 4,
    TypeUint32WrongSize )
```

12.86.4.4 CompileTimeAssert() [4/9]

```
CompileTimeAssert (
    sizeof(uint64) == 8,
    TypeUint64WrongSize )
```

12.86.4.5 CompileTimeAssert() [5/9]

```
CompileTimeAssert (
    sizeof(int8)  == 1,
    Typeint8WrongSize )
```

12.86.4.6 CompileTimeAssert() [6/9]

```
CompileTimeAssert (
    sizeof(int16) == 2,
    Typeint16WrongSize )
```

12.86.4.7 CompileTimeAssert() [7/9]

```
CompileTimeAssert (
    sizeof(int32) == 4,
    Typeint32WrongSize )
```

12.86.4.8 CompileTimeAssert() [8/9]

```
CompileTimeAssert (
    sizeof(int64) == 8,
    Typeint64WrongSize )
```

12.86.4.9 CompileTimeAssert() [9/9]

```
CompileTimeAssert (
    sizeof(cpuaddr) >= sizeof(void *) ,
    TypePtrWrongSize )
```

12.87 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.

12.87.1 Detailed Description

Declarations and prototypes for binary semaphores

12.88 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)`

12.88.1 Detailed Description

Declarations and prototypes for OSAL BSP

12.89 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.

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.
- static `int64 OS_TimeGetTotalSeconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to whole number of seconds.
- static `int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to millisecond units.
- static `int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to microsecond units.
- static `int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`
Get interval from an `OS_time_t` object normalized to nanosecond units.
- 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)`
Get microseconds portion (fractional part only) from an `OS_time_t` object.
- static `uint32 OS_TimeGetNanosecondsPart (OS_time_t tm)`
Get nanoseconds portion (fractional part only) from an `OS_time_t` object.

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.

12.89.1 Detailed Description

Declarations and prototypes for osapi-clock module

12.89.2 Enumeration Type Documentation

12.89.2.1 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

<code>OS_TIME_TICK_RESOLUTION_NS</code>	
<code>OS_TIME_TICKS_PER_SECOND</code>	
<code>OS_TIME_TICKS_PER_MSEC</code>	
<code>OS_TIME_TICKS_PER_USEC</code>	

Definition at line 61 of file osapi-clock.h.

12.90 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.

12.90.1 Detailed Description

Declarations and prototypes for general OSAL functions that are not part of a subsystem

12.90.2 Typedef Documentation

12.90.2.1 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

in	<i>event</i>	The event that occurred
in	<i>object_id</i>	The associated object_id, or 0 if not associated with an object
in, out	<i>data</i>	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.

12.90.3 Enumeration Type Documentation**12.90.3.1 OS_Event_t**

enum [OS_Event_t](#)

A set of events that can be used with BSP event callback routines.

Enumerator

OS_EVENT_RESERVED	no-op/reserved event id value
OS_EVENT_RESOURCE_ALLOCATED	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.
OS_EVENT_RESOURCE_CREATED	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.
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.

12.91 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_ForEachObjectOfType() 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.

12.91.1 Detailed Description

General constants for OSAL that are shared across subsystems

12.91.2 Macro Definition Documentation

12.91.2.1 OS_CHECK

```
#define OS_CHECK (0)
```

Definition at line 35 of file osapi-constants.h.

12.91.2.2 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.

12.91.2.3 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.

12.91.2.4 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.

12.91.2.5 OS_PEND

```
#define OS_PEND (-1)
```

Definition at line 34 of file osapi-constants.h.

12.92 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.

12.92.1 Detailed Description

Declarations and prototypes for counting semaphores

12.93 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.

12.93.1 Detailed Description

Declarations and prototypes for directories

12.93.2 Macro Definition Documentation

12.93.2.1 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.

12.94 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_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.

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.

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.

12.94.1 Detailed Description

OSAL error code definitions

12.94.2 Macro Definition Documentation

12.94.2.1 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`.

12.94.3 Typedef Documentation

12.94.3.1 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.

12.95 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_TimedRead (osal_id_t filedes, void *buffer, size_t nbytes, int32 timeout)`
File/Stream input read 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.

12.95.1 Detailed Description

Declarations and prototypes for file objects

12.95.2 Macro Definition Documentation

12.95.2.1 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.

12.95.2.2 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.

12.95.2.3 OS_FILESTAT_MODE

```
#define OS_FILESTAT_MODE( x ) ((x). FileModeBits)
```

Access file stat mode bits.

Definition at line 88 of file osapi-file.h.

12.95.2.4 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.

12.95.2.5 OS_FILESTAT_SIZE

```
#define OS_FILESTAT_SIZE(
    x ) ((x).FileSize)
```

Access file stat size field.

Definition at line 98 of file osapi-file.h.

12.95.2.6 OS_FILESTAT_TIME

```
#define OS_FILESTAT_TIME(
    x ) (OS_TimeGetTotalSeconds((x).FileTime))
```

Access file stat time field as a whole number of seconds.

Definition at line 100 of file osapi-file.h.

12.95.2.7 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.

12.95.3 Enumeration Type Documentation

12.95.3.1 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.

12.95.3.2 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.

12.96 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 *filesystem_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.

12.96.1 Detailed Description

Declarations and prototypes for file systems

12.96.2 Macro Definition Documentation

12.96.2.1 OS_CHK_ONLY

```
#define OS_CHK_ONLY 0
```

Unused, API takes bool

Definition at line 31 of file osapi-filesystems.h.

12.96.2.2 OS_REPAIR

```
#define OS_REPAIR 1
```

Unused, API takes bool

Definition at line 32 of file osapi-filesystems.h.

12.97 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.

12.97.1 Detailed Description

Declarations and prototypes for heap functions

12.98 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

- `#define OS_OBJECT_TYPE_OS_STREAM 0x06`
Object mutex type.
- `#define OS_OBJECT_TYPE_OS_DIR 0x07`
Object stream type.
- `#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08`
Object directory type.
- `#define OS_OBJECT_TYPE_OS_TIMECB 0x09`
Object timebase type.
- `#define OS_OBJECT_TYPE_OS_MODULE 0x0A`
Object timer callback type.
- `#define OS_OBJECT_TYPE_OS_FILESYS 0x0B`
Object module type.
- `#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C`
Object file system type.
- `#define OS_OBJECT_TYPE_USER 0x10`
Object console 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_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

12.98.1 Detailed Description

Declarations and prototypes for object IDs

12.98.2 Macro Definition Documentation

12.98.2.1 OS_OBJECT_INDEX_MASK

```
#define OS_OBJECT_INDEX_MASK 0xFFFF
```

Object index mask.

Definition at line 32 of file osapi-idmap.h.

12.98.2.2 OS_OBJECT_TYPE_SHIFT

```
#define OS_OBJECT_TYPE_SHIFT 16
```

Object type shift.

Definition at line 33 of file osapi-idmap.h.

12.99 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.

12.99.1 Detailed Description

Macro definitions that are used across all OSAL subsystems

12.99.2 Macro Definition Documentation

12.99.2.1 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 122 of file osapi-macros.h.

12.99.2.2 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 96 of file osapi-macros.h.

12.99.2.3 BUGREPORT

```
#define BUGREPORT( ... ) OS_printf(__VA_ARGS__)
```

Definition at line 79 of file osapi-macros.h.

12.99.2.4 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 137 of file osapi-macros.h.

12.100 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.

12.100.1 Detailed Description

Declarations and prototypes for module subsystem

12.100.2 Macro Definition Documentation

12.100.2.1 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.

12.100.2.2 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.

12.101 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.

12.101.1 Detailed Description

Declarations and prototypes for mutexes

12.102 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.

12.102.1 Detailed Description

Declarations and prototypes for network subsystem

12.103 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.

12.103.1 Detailed Description

Declarations and prototypes for printf/console output

12.104 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.

12.104.1 Detailed Description

Declarations and prototypes for queue subsystem

12.105 osal/src/os/inc/osapi-select.h File Reference

```
#include "osconfig.h"
#include "common_types.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 }

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_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)`
Wait for events across multiple file handles.
- `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_SelectFdIsSet (const OS_FdSet *Set, osal_id_t objid)`
Check if an FdSet structure contains a given ID.

12.105.1 Detailed Description

Declarations and prototypes for select abstraction

12.105.2 Enumeration Type Documentation

12.105.2.1 `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

<code>OS_STREAM_STATE_BOUND</code>	whether the stream is bound
<code>OS_STREAM_STATE_CONNECTED</code>	whether the stream is connected
<code>OS_STREAM_STATE_READABLE</code>	whether the stream is readable
<code>OS_STREAM_STATE_WRITABLE</code>	whether the stream is writable

Definition at line 55 of file osapi-select.h.

12.106 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.

12.106.1 Detailed Description

Declarations and prototypes for shell abstraction

12.107 osal/src/os/inc/osapi-sockets.h File Reference

```
#include "osconfig.h"
#include "common_types.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.
- `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_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_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.

12.107.1 Detailed Description

Declarations and prototypes for sockets abstraction

12.107.2 Macro Definition Documentation

12.107.2.1 OS SOCKADDR_MAX_LEN

```
#define OS_SOCKADDR_MAX_LEN 28
```

Definition at line 45 of file osapi-sockets.h.

12.107.3 Enumeration Type Documentation

12.107.3.1 OS_SocketDomain_t

```
enum OS_SocketDomain_t
```

Socket domain.

Enumerator

OS_SocketDomain_INVALID	Invalid.
OS_SocketDomain_INET	IPv4 address family, most commonly used)
OS_SocketDomain_INET6	IPv6 address family, depends on OS/network stack support.
OS_SocketDomain_MAX	Maximum.

Definition at line 60 of file osapi-sockets.h.

12.107.3.2 OS_SocketShutdownMode_t

```
enum OS_SocketShutdownMode_t
```

Shutdown Mode.

Enumerator

OS_SocketShutdownMode_NONE	Reserved value, no effect.
OS_SocketShutdownMode_SHUT_READ	Disable future reading.
OS_SocketShutdownMode_SHUT_WRITE	Disable future writing.
OS_SocketShutdownMode_SHUT_RDWR	Disable future reading or writing.

Definition at line 79 of file osapi-sockets.h.

12.107.3.3 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 69 of file osapi-sockets.h.

12.108 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.

12.108.1 Detailed Description

Declarations and prototypes for task abstraction

12.108.2 Macro Definition Documentation

12.108.2.1 OS_FP_ENABLED

```
#define OS_FP_ENABLED 1
```

Floating point enabled state for a task.

Definition at line 35 of file osapi-task.h.

12.108.2.2 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.

12.108.2.3 OSAL_PRIORITY_C

```
#define OSAL_PRIORITY_C(  
    X ) ((osal_priority_t) {X})
```

Definition at line 46 of file osapi-task.h.

12.108.2.4 OSAL_STACKPTR_C

```
#define OSAL_STACKPTR_C(  
    X ) ((osal_stackptr_t) {X})
```

Definition at line 53 of file osapi-task.h.

12.108.2.5 OSAL_TASK_STACK_ALLOCATE

```
#define OSAL_TASK_STACK_ALLOCATE OSAL_STACKPTR_C(NULL)
```

Definition at line 54 of file osapi-task.h.

12.108.3 Typedef Documentation

12.108.3.1 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.

12.108.3.2 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.

12.108.3.3 osal_task

```
typedef void osal_task
```

For task entry point.

Definition at line 68 of file osapi-task.h.

12.108.4 Function Documentation

12.108.4.1 osal_task()

```
typedef osal_task (
    (*) (void) osal_task_entry )
```

For task entry point.

12.109 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.

12.109.1 Detailed Description

Declarations and prototypes for timebase abstraction

12.109.2 Typedef Documentation

12.109.2.1 OS_TimerSync_t

```
typedef uint32 (* OS_TimerSync_t) (osal_id_t timer_id)
```

Timer sync.

Definition at line 34 of file osapi-timebase.h.

12.110 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.

12.110.1 Detailed Description

Declarations and prototypes for timer abstraction (app callbacks)

12.110.2 Typedef Documentation

12.110.2.1 OS_TimerCallback_t

```
typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)
```

Timer callback.

Definition at line 34 of file osapi-timer.h.

12.111 osal/src/os/inc/osapi-version.h File Reference

```
#include "common_types.h"
```

Macros

- `#define OS_BUILD_NUMBER 83`
- `#define OS_BUILD_BASELINE "v6.0.0-rc4"`
- `#define OS_MAJOR_VERSION 5`
Major version number.
- `#define OS_MINOR_VERSION 0`
Minor version number.
- `#define OS_REVISION 99`
Revision version number. Value of 99 indicates a development version.
- `#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 OS_VERSION_CODENAME "Draco"`
Version code name All modular components which are tested/validated together should share the same code name.
- `#define OS_VERSION_STRING`
Development Build Version String.
- `#define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION * 100) + OS_REVISION)`
Combines the revision components into a single value.

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.

12.111.1 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.

12.111.2 Macro Definition Documentation

12.111.2.1 OS_BUILD_BASELINE

```
#define OS_BUILD_BASELINE "v6.0.0-rc4"
```

Definition at line 38 of file osapi-version.h.

12.111.2.2 OS_BUILD_NUMBER

```
#define OS_BUILD_NUMBER 83
```

Definition at line 37 of file osapi-version.h.

12.111.2.3 OS_MAJOR_VERSION

```
#define OS_MAJOR_VERSION 5
```

Major version number.

Definition at line 43 of file osapi-version.h.

12.111.2.4 OS_MINOR_VERSION

```
#define OS_MINOR_VERSION 0
```

Minor version number.

Definition at line 44 of file osapi-version.h.

12.111.2.5 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 54 of file osapi-version.h.

12.111.2.6 OS_REVISION

```
#define OS_REVISION 99
```

Revision version number. Value of 99 indicates a development version.

Definition at line 45 of file osapi-version.h.

12.111.2.7 OS_STR

```
#define OS_STR( x ) OS_STR_HELPER(x)
```

Helper function to concatenate strings from integer.

Definition at line 60 of file osapi-version.h.

12.111.2.8 OS_STR_HELPER

```
#define OS_STR_HELPER( x ) #x
```

Helper function to concatenate strings from integer.

Definition at line 59 of file osapi-version.h.

12.111.2.9 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 65 of file osapi-version.h.

12.111.2.10 OS_VERSION_CODENAME

```
#define OS_VERSION_CODENAME "Draco"
```

Version code name All modular components which are tested/validated together should share the same code name.

Definition at line 70 of file osapi-version.h.

12.111.2.11 OS_VERSION_STRING

```
#define OS_VERSION_STRING
```

Value:

```
" OSAL Development Build\n"
" " OS_VERSION " (Codename: " OS_VERSION_CODENAME ") \n" /* Codename for
current development */ \
" Latest Official Version: osal v5.0.0" \
/* For full support please use official release
version */ \\\
```

Development Build Version String.

Reports the current development build's baseline, number, and name. Also includes a note about the latest official version.

Definition at line 76 of file osapi-version.h.

12.111.2.12 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 86 of file osapi-version.h.

12.111.3 Function Documentation

12.111.3.1 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

12.111.3.2 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.

12.111.3.3 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

out	<i>VersionNumbers</i>	A fixed-size array to be filled with the version numbers
-----	-----------------------	--

12.111.3.4 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.

12.112 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-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"
```

12.112.1 Detailed Description

Purpose: Contains functions prototype definitions and variables declarations for the OS Abstraction Layer, Core OS module

12.113 psp/fsw/inc/cfe_psp.h File Reference

```
#include "common_types.h"
#include "osapi.h"
```

Macros

- #define CFE_PSP_SUCCESS (0)
- #define CFE_PSP_ERROR (-1)
- #define CFE_PSP_INVALID_POINTER (-2)
- #define CFE_PSP_ERROR_ADDRESS_MISALIGNED (-3)
- #define CFE_PSP_ERROR_TIMEOUT (-4)
- #define CFE_PSP_INVALID_INT_NUM (-5)
- #define CFE_PSP_INVALID_MEM_ADDR (-21)
- #define CFE_PSP_INVALID_MEM_TYPE (-22)
- #define CFE_PSP_INVALID_MEM_RANGE (-23)
- #define CFE_PSP_INVALID_MEM_WORDSIZE (-24)
- #define CFE_PSP_INVALID_MEM_SIZE (-25)
- #define CFE_PSP_INVALID_MEM_ATTR (-26)
- #define CFE_PSP_ERROR_NOT_IMPLEMENTED (-27)
- #define CFE_PSP_INVALID_MODULE_NAME (-28)
- #define CFE_PSP_INVALID_MODULE_ID (-29)
- #define CFE_PSP_NO_EXCEPTION_DATA (-30)
- #define CFE_PSP_PANIC_STARTUP 1
- #define CFE_PSP_PANIC_VOLATILE_DISK 2
- #define CFE_PSP_PANIC_MEMORY_ALLOC 3
- #define CFE_PSP_PANIC_NONVOL_DISK 4
- #define CFE_PSP_PANIC_STARTUP_SEM 5
- #define CFE_PSP_PANIC_CORE_APP 6
- #define CFE_PSP_PANIC_GENERAL_FAILURE 7
- #define BUFF_SIZE 256
- #define SIZE_BYTE 1
- #define SIZE_HALF 2
- #define SIZE_WORD 3
- #define CFE_PSP_MEM_RAM 1
- #define CFE_PSP_MEM_EEPROM 2
- #define CFE_PSP_MEM_ANY 3
- #define CFE_PSP_MEM_INVALID 4
- #define CFE_PSP_MEM_ATTR_WRITE 0x01
- #define CFE_PSP_MEM_ATTR_READ 0x02
- #define CFE_PSP_MEM_ATTR_READWRITE 0x03
- #define CFE_PSP_MEM_SIZE_BYTE 0x01
- #define CFE_PSP_MEM_SIZE_WORD 0x02
- #define CFE_PSP_MEM_SIZE_DWORD 0x04
- #define CFE_PSP_SOFT_TIMEBASE_NAME "cFS-Master"

The name of the software/RTOS timebase for general system timers.

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

- `#define CFE_PSP_RST_SUBTYPE_PUSH_BUTTON 2`
Reset caused by power having been removed and restored.
 - `#define CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND 3`
Reset caused by reset button on the board.
 - `#define CFE_PSP_RST_SUBTYPE_HW_WATCHDOG 4`
Reset was caused by a reset line having been stimulated by a hardware special command.
 - `#define CFE_PSP_RST_SUBTYPE_RESET_COMMAND 5`
Reset was caused by a watchdog timer expiring.
 - `#define CFE_PSP_RST_SUBTYPE_EXCEPTION 6`
Reset was caused by cFE ES processing a [Reset Command](#).
 - `#define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7`
Reset was caused by a Processor Exception.
 - `#define CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET 8`
Reset was caused in an unknown manner.
 - `#define CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET 9`
Reset was caused by a JTAG or BDM connection.
 - `#define CFE_PSP_RST_SUBTYPE_MAX 10`
Reset reverted to a cFE POWERON due to a boot bank switch.
- Placeholder to indicate 1+ the maximum value that the PSP will ever use.*

Functions

- `void CFE_PSP_Main (void)`
- `void CFE_PSP_GetTime (OS_time_t *LocalTime)`
Sample/Read a monotonic platform clock with normalization.
- `void CFE_PSP_Restart (uint32 resetType)`
- `uint32 CFE_PSP_GetRestartType (uint32 *restartSubType)`
- `void CFE_PSP_FlushCaches (uint32 type, void *address, uint32 size)`
- `uint32 CFE_PSP_GetProcessorId (void)`
- `uint32 CFE_PSP_GetSpacecraftId (void)`
- `const char * CFE_PSP_GetProcessorName (void)`
- `uint32 CFE_PSP_Get_Timer_Tick (void)`
- `uint32 CFE_PSP_GetTimerTicksPerSecond (void)`
- `uint32 CFE_PSP_GetTimerLow32Rollover (void)`
- `void CFE_PSP_Get_Timebase (uint32 *Tbu, uint32 *Tbl)`
Sample/Read a monotonic platform clock without normalization.
- `uint32 CFE_PSP_Get_Dec (void)`
- `int32 CFE_PSP_GetCDSSize (uint32 *SizeOfCDS)`
- `int32 CFE_PSP_WriteToCDS (const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)`
- `int32 CFE_PSP_ReadFromCDS (void *PtrToDataToRead, uint32 CDSOffset, uint32 NumBytes)`
- `int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)`
- `int32 CFE_PSP.GetUserReservedArea (cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)`
- `int32 CFE_PSP_GetVolatileDiskMem (cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)`
- `int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)`
- `int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)`
- `void CFE_PSP_WatchdogInit (void)`
- `void CFE_PSP_WatchdogEnable (void)`
- `void CFE_PSP_WatchdogDisable (void)`
- `void CFE_PSP_WatchdogService (void)`
- `uint32 CFE_PSP_WatchdogGet (void)`
- `void CFE_PSP_WatchdogSet (uint32 WatchdogValue)`

- void `CFE_PSP_Panic` (`int32 ErrorCode`)
- `int32 CFE_PSP_InitSSR` (`uint32 bus`, `uint32 device`, `char *DeviceName`)
- `int32 CFE_PSP_Decompress` (`char *srcFileName`, `char *dstFileName`)
- void `CFE_PSP_AttachExceptions` (`void`)
- void `CFE_PSP_SetDefaultExceptionEnvironment` (`void`)
- `uint32 CFE_PSP_Exception_GetCount` (`void`)
- `int32 CFE_PSP_Exception_GetSummary` (`uint32 *ContextLogId`, `osal_id_t *TaskId`, `char *ReasonBuf`, `uint32 ReasonSize`)
- `int32 CFE_PSP_Exception_CopyContext` (`uint32 ContextLogId`, `void *ContextBuf`, `uint32 ContextSize`)
- `int32 CFE_PSP_PortRead8` (`cpuaddr PortAddress`, `uint8 *ByteValue`)
- `int32 CFE_PSP_PortWrite8` (`cpuaddr PortAddress`, `uint8 ByteValue`)
- `int32 CFE_PSP_PortRead16` (`cpuaddr PortAddress`, `uint16 *uint16Value`)
- `int32 CFE_PSP_PortWrite16` (`cpuaddr PortAddress`, `uint16 uint16Value`)
- `int32 CFE_PSP_PortRead32` (`cpuaddr PortAddress`, `uint32 *uint32Value`)
- `int32 CFE_PSP_PortWrite32` (`cpuaddr PortAddress`, `uint32 uint32Value`)
- `int32 CFE_PSP_MemRead8` (`cpuaddr MemoryAddress`, `uint8 *ByteValue`)
- `int32 CFE_PSP_MemWrite8` (`cpuaddr MemoryAddress`, `uint8 ByteValue`)
- `int32 CFE_PSP_MemRead16` (`cpuaddr MemoryAddress`, `uint16 *uint16Value`)
- `int32 CFE_PSP_MemWrite16` (`cpuaddr MemoryAddress`, `uint16 uint16Value`)
- `int32 CFE_PSP_MemRead32` (`cpuaddr MemoryAddress`, `uint32 *uint32Value`)
- `int32 CFE_PSP_MemWrite32` (`cpuaddr MemoryAddress`, `uint32 uint32Value`)
- `int32 CFE_PSP_MemCpy` (`void *dest`, `const void *src`, `uint32 n`)
- `int32 CFE_PSP_MemSet` (`void *dest`, `uint8 value`, `uint32 n`)
- `int32 CFE_PSP_MemValidateRange` (`cpuaddr Address`, `size_t Size`, `uint32 MemoryType`)
- `uint32 CFE_PSP_MemRanges` (`void`)
- `int32 CFE_PSP_MemRangeSet` (`uint32 RangeNum`, `uint32 MemoryType`, `cpuaddr StartAddr`, `size_t Size`, `size_t WordSize`, `uint32 Attributes`)
- `int32 CFE_PSP_MemRangeGet` (`uint32 RangeNum`, `uint32 *MemoryType`, `cpuaddr *StartAddr`, `size_t *Size`, `size_t *WordSize`, `uint32 *Attributes`)
- `int32 CFE_PSP_EepromWrite8` (`cpuaddr MemoryAddress`, `uint8 ByteValue`)
- `int32 CFE_PSP_EepromWrite16` (`cpuaddr MemoryAddress`, `uint16 uint16Value`)
- `int32 CFE_PSP_EepromWrite32` (`cpuaddr MemoryAddress`, `uint32 uint32Value`)
- `int32 CFE_PSP_EepromWriteEnable` (`uint32 Bank`)
- `int32 CFE_PSP_EepromWriteDisable` (`uint32 Bank`)
- `int32 CFE_PSP_EepromPowerUp` (`uint32 Bank`)
- `int32 CFE_PSP_EepromPowerDown` (`uint32 Bank`)
- `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]`)

Obtain the PSP numeric version numbers as uint8 values.
- `uint32 CFE_PSP_GetBuildNumber` (`void`)

Obtain the PSP library numeric build number.

12.113.1 Macro Definition Documentation

12.113.1.1 BUFF_SIZE

```
#define BUFF_SIZE 256
```

Definition at line 80 of file cfe_psp.h.

12.113.1.2 CFE_PSP_ERROR

```
#define CFE_PSP_ERROR (-1)
```

Definition at line 50 of file cfe_psp.h.

12.113.1.3 CFE_PSP_ERROR_ADDRESS_MISALIGNED

```
#define CFE_PSP_ERROR_ADDRESS_MISALIGNED (-3)
```

Definition at line 52 of file cfe_psp.h.

12.113.1.4 CFE_PSP_ERROR_NOT_IMPLEMENTED

```
#define CFE_PSP_ERROR_NOT_IMPLEMENTED (-27)
```

Definition at line 61 of file cfe_psp.h.

12.113.1.5 CFE_PSP_ERROR_TIMEOUT

```
#define CFE_PSP_ERROR_TIMEOUT (-4)
```

Definition at line 53 of file cfe_psp.h.

12.113.1.6 CFE_PSP_INVALID_INT_NUM

```
#define CFE_PSP_INVALID_INT_NUM (-5)
```

Definition at line 54 of file cfe_psp.h.

12.113.1.7 CFE_PSP_INVALID_MEM_ADDR

```
#define CFE_PSP_INVALID_MEM_ADDR (-21)
```

Definition at line 55 of file cfe_psp.h.

12.113.1.8 CFE_PSP_INVALID_MEM_ATTR

```
#define CFE_PSP_INVALID_MEM_ATTR (-26)
```

Definition at line 60 of file cfe_psp.h.

12.113.1.9 CFE_PSP_INVALID_MEM_RANGE

```
#define CFE_PSP_INVALID_MEM_RANGE (-23)
```

Definition at line 57 of file cfe_psp.h.

12.113.1.10 CFE_PSP_INVALID_MEM_SIZE

```
#define CFE_PSP_INVALID_MEM_SIZE (-25)
```

Definition at line 59 of file cfe_psp.h.

12.113.1.11 CFE_PSP_INVALID_MEM_TYPE

```
#define CFE_PSP_INVALID_MEM_TYPE (-22)
```

Definition at line 56 of file cfe_psp.h.

12.113.1.12 CFE_PSP_INVALID_MEM_WORDSIZE

```
#define CFE_PSP_INVALID_MEM_WORDSIZE (-24)
```

Definition at line 58 of file cfe_psp.h.

12.113.1.13 CFE_PSP_INVALID_MODULE_ID

```
#define CFE_PSP_INVALID_MODULE_ID (-29)
```

Definition at line 63 of file cfe_psp.h.

12.113.1.14 CFE_PSP_INVALID_MODULE_NAME

```
#define CFE_PSP_INVALID_MODULE_NAME (-28)
```

Definition at line 62 of file cfe_psp.h.

12.113.1.15 CFE_PSP_INVALID_POINTER

```
#define CFE_PSP_INVALID_POINTER (-2)
```

Definition at line 51 of file cfe_psp.h.

12.113.1.16 CFE_PSP_MEM_ANY

```
#define CFE_PSP_MEM_ANY 3
```

Definition at line 90 of file cfe_psp.h.

12.113.1.17 CFE_PSP_MEM_ATTR_READ

```
#define CFE_PSP_MEM_ATTR_READ 0x02
```

Definition at line 97 of file cfe_psp.h.

12.113.1.18 CFE_PSP_MEM_ATTR_READWRITE

```
#define CFE_PSP_MEM_ATTR_READWRITE 0x03
```

Definition at line 98 of file cfe_psp.h.

12.113.1.19 CFE_PSP_MEM_ATTR_WRITE

```
#define CFE_PSP_MEM_ATTR_WRITE 0x01
```

Definition at line 96 of file cfe_psp.h.

12.113.1.20 CFE_PSP_MEM_EEPROM

```
#define CFE_PSP_MEM_EEPROM 2
```

Definition at line 89 of file cfe_psp.h.

12.113.1.21 CFE_PSP_MEM_INVALID

```
#define CFE_PSP_MEM_INVALID 4
```

Definition at line 91 of file cfe_psp.h.

12.113.1.22 CFE_PSP_MEM_RAM

```
#define CFE_PSP_MEM_RAM 1
```

Definition at line 88 of file cfe_psp.h.

12.113.1.23 CFE_PSP_MEM_SIZE_BYTE

```
#define CFE_PSP_MEM_SIZE_BYTE 0x01
```

Definition at line 103 of file cfe_psp.h.

12.113.1.24 CFE_PSP_MEM_SIZE_DWORD

```
#define CFE_PSP_MEM_SIZE_DWORD 0x04
```

Definition at line 105 of file cfe_psp.h.

12.113.1.25 CFE_PSP_MEM_SIZE_WORD

```
#define CFE_PSP_MEM_SIZE_WORD 0x02
```

Definition at line 104 of file cfe_psp.h.

12.113.1.26 CFE_PSP_NO_EXCEPTION_DATA

```
#define CFE_PSP_NO_EXCEPTION_DATA (-30)
```

Definition at line 64 of file cfe_psp.h.

12.113.1.27 CFE_PSP_PANIC_CORE_APP

```
#define CFE_PSP_PANIC_CORE_APP 6
```

Definition at line 74 of file cfe_psp.h.

12.113.1.28 CFE_PSP_PANIC_GENERAL_FAILURE

```
#define CFE_PSP_PANIC_GENERAL_FAILURE 7
```

Definition at line 75 of file cfe_psp.h.

12.113.1.29 CFE_PSP_PANIC_MEMORY_ALLOC

```
#define CFE_PSP_PANIC_MEMORY_ALLOC 3
```

Definition at line 71 of file cfe_psp.h.

12.113.1.30 CFE_PSP_PANIC_NONVOL_DISK

```
#define CFE_PSP_PANIC_NONVOL_DISK 4
```

Definition at line 72 of file cfe_psp.h.

12.113.1.31 CFE_PSP_PANIC_STARTUP

```
#define CFE_PSP_PANIC_STARTUP 1
```

Definition at line 69 of file cfe_psp.h.

12.113.1.32 CFE_PSP_PANIC_STARTUP_SEM

```
#define CFE_PSP_PANIC_STARTUP_SEM 5
```

Definition at line 73 of file cfe_psp.h.

12.113.1.33 CFE_PSP_PANIC_VOLATILE_DISK

```
#define CFE_PSP_PANIC_VOLATILE_DISK 2
```

Definition at line 70 of file cfe_psp.h.

12.113.1.34 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 140 of file cfe_psp.h.

12.113.1.35 CFE_PSP_RST_SUBTYPE_EXCEPTION

```
#define CFE_PSP_RST_SUBTYPE_EXCEPTION 6
```

Reset was caused by a Processor Exception.

Definition at line 134 of file cfe_psp.h.

12.113.1.36 CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND

```
#define CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND 3
```

Reset was caused by a reset line having been stimulated by a hardware special command.

Definition at line 128 of file cfe_psp.h.

12.113.1.37 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 130 of file cfe_psp.h.

12.113.1.38 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 138 of file cfe_psp.h.

12.113.1.39 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 142 of file cfe_psp.h.

12.113.1.40 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 124 of file cfe_psp.h.

12.113.1.41 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 126 of file cfe_psp.h.

12.113.1.42 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 132 of file `cfe_psp.h`.

12.113.1.43 CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET

```
#define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7
```

Reset was caused in an unknown manner.

Definition at line 136 of file `cfe_psp.h`.

12.113.1.44 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 114 of file `cfe_psp.h`.

12.113.1.45 CFE_PSP_RST_TYPE_POWERON

```
#define CFE_PSP_RST_TYPE_POWERON 2
```

All memory has been cleared

Definition at line 113 of file `cfe_psp.h`.

12.113.1.46 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 112 of file `cfe_psp.h`.

Referenced by `HS_MonitorApplications()`, and `HS_MonitorEvent()`.

12.113.1.47 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 150 of file cfe_psp.h.

12.113.1.48 CFE_PSP_SUCCESS

```
#define CFE_PSP_SUCCESS (0)
```

Definition at line 49 of file cfe_psp.h.

12.113.1.49 SIZE_BYTE

```
#define SIZE_BYTE 1
```

Definition at line 81 of file cfe_psp.h.

12.113.1.50 SIZE_HALF

```
#define SIZE_HALF 2
```

Definition at line 82 of file cfe_psp.h.

12.113.1.51 SIZE_WORD

```
#define SIZE_WORD 3
```

Definition at line 83 of file cfe_psp.h.

12.113.2 Function Documentation

12.113.2.1 CFE_PSP_AttachExceptions()

```
void CFE_PSP_AttachExceptions (
    void )
```

12.113.2.2 CFE_PSP_Decompress()

```
int32 CFE_PSP_Decompress (
    char * srcFileName,
    char * dstFileName )
```

12.113.2.3 CFE_PSP_EepromPowerDown()

```
int32 CFE_PSP_EepromPowerDown (
    uint32 Bank )
```

12.113.2.4 CFE_PSP_EepromPowerUp()

```
int32 CFE_PSP_EepromPowerUp (
    uint32 Bank )
```

12.113.2.5 CFE_PSP_EepromWrite16()

```
int32 CFE_PSP_EepromWrite16 (
    cpuaddr MemoryAddress,
    uint16 uint16Value )
```

12.113.2.6 CFE_PSP_EepromWrite32()

```
int32 CFE_PSP_EepromWrite32 (
    cpuaddr MemoryAddress,
    uint32 uint32Value )
```

12.113.2.7 CFE_PSP_EepromWrite8()

```
int32 CFE_PSP_EepromWrite8 (
    cpuaddr MemoryAddress,
    uint8 ByteValue )
```

12.113.2.8 CFE_PSP_EepromWriteDisable()

```
int32 CFE_PSP_EepromWriteDisable (
    uint32 Bank )
```

12.113.2.9 CFE_PSP_EepromWriteEnable()

```
int32 CFE_PSP_EepromWriteEnable (
    uint32 Bank )
```

12.113.2.10 CFE_PSP_Exception_CopyContext()

```
int32 CFE_PSP_Exception_CopyContext (
    uint32 ContextLogId,
    void * ContextBuf,
    uint32 ContextSize )
```

12.113.2.11 CFE_PSP_Exception_GetCount()

```
uint32 CFE_PSP_Exception_GetCount (
    void )
```

12.113.2.12 CFE_PSP_Exception_GetSummary()

```
int32 CFE_PSP_Exception_GetSummary (
    uint32 * ContextLogId,
    osal_id_t * TaskId,
    char * ReasonBuf,
    uint32 ReasonSize )
```

12.113.2.13 CFE_PSP_FlushCaches()

```
void CFE_PSP_FlushCaches (
    uint32 type,
    void * address,
    uint32 size )
```

12.113.2.14 CFE_PSP_Get_Dec()

```
uint32 CFE_PSP_Get_Dec (
    void )
```

12.113.2.15 CFE_PSP_Get_Timebase()

```
void CFE_PSP_Get_Timebase (
    uint32 * Tbu,
    uint32 * Tbl )
```

Sample/Read a monotonic platform clock without normalization.

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	<i>Tbu</i>	Buffer to hold the upper 32 bits of a 64-bit tick counter
out	<i>Tbl</i>	Buffer to hold the lower 32 bits of a 64-bit tick counter

12.113.2.16 CFE_PSP_Get_Timer_Tick()

```
uint32 CFE_PSP_Get_Timer_Tick (
    void )
```

12.113.2.17 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

12.113.2.18 CFE_PSP_GetCDSSize()

```
int32 CFE_PSP_GetCDSSize (
    uint32 * SizeOfCDS )
```

12.113.2.19 CFE_PSP_GetCFETextSegmentInfo()

```
int32 CFE_PSP_GetCFETextSegmentInfo (
    cpuaddr * PtrToCFEsegment,
    uint32 * SizeOfCFEsegment )
```

12.113.2.20 CFE_PSP_GetKernelTextSegmentInfo()

```
int32 CFE_PSP_GetKernelTextSegmentInfo (
    cpuaddr * PtrToKernelSegment,
    uint32 * SizeOfKernelSegment )
```

12.113.2.21 CFE_PSP_GetProcessorId()

```
uint32 CFE_PSP_GetProcessorId (
    void )
```

12.113.2.22 CFE_PSP_GetProcessorName()

```
const char* CFE_PSP_GetProcessorName (
    void )
```

12.113.2.23 CFE_PSP_GetResetArea()

```
int32 CFE_PSP_GetResetArea (
    cpuaddr * PtrToResetArea,
    uint32 * SizeOfResetArea )
```

12.113.2.24 CFE_PSP_GetRestartType()

```
uint32 CFE_PSP_GetRestartType (
    uint32 * restartSubType )
```

12.113.2.25 CFE_PSP_GetSpacecraftId()

```
uint32 CFE_PSP_GetSpacecraftId (
    void )
```

12.113.2.26 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 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 <code>OS_time_t</code>
------------------	------------------------	--

Referenced by HS_IdleTask().

12.113.2.27 CFE_PSP_GetTimerLow32Rollover()

```
uint32 CFE_PSP_GetTimerLow32Rollover (
    void )
```

12.113.2.28 CFE_PSP_GetTimerTicksPerSecond()

```
uint32 CFE_PSP_GetTimerTicksPerSecond (
    void )
```

12.113.2.29 CFE_PSP.GetUserReservedArea()

```
int32 CFE_PSP.GetUserReservedArea (
    cpuaddr * PtrToUserArea,
    uint32 * SizeOfUserArea )
```

12.113.2.30 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.

12.113.2.31 CFE_PSP_GetVersionNumber()

```
void CFE_PSP_GetVersionNumber (
    uint8 VersionNumbers[4] )
```

Obtain the PSP numeric version numbers as uint8 values.

This retrieves the numeric PSP 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
------------------	-----------------------------	--

12.113.2.32 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.

12.113.2.33 CFE_PSP_GetVolatileDiskMem()

```
int32 CFE_PSP_GetVolatileDiskMem (
    cpuaddr * PtrToVolDisk,
    uint32 * SizeOfVolDisk )
```

12.113.2.34 CFE_PSP_InitSSR()

```
int32 CFE_PSP_InitSSR (
    uint32 bus,
    uint32 device,
    char * DeviceName )
```

12.113.2.35 CFE_PSP_Main()

```
void CFE_PSP_Main (
    void )
```

12.113.2.36 CFE_PSP_MemCpy()

```
int32 CFE_PSP_MemCpy (
    void * dest,
    const void * src,
    uint32 n )
```

12.113.2.37 CFE_PSP_MemRangeGet()

```
int32 CFE_PSP_MemRangeGet (
    uint32 RangeNum,
    uint32 * MemoryType,
    cpuaddr * StartAddr,
    size_t * Size,
    size_t * WordSize,
    uint32 * Attributes )
```

12.113.2.38 CFE_PSP_MemRanges()

```
uint32 CFE_PSP_MemRanges (
    void )
```

12.113.2.39 CFE_PSP_MemRangeSet()

```
int32 CFE_PSP_MemRangeSet (
    uint32 RangeNum,
    uint32 MemoryType,
    cpuaddr StartAddr,
    size_t Size,
    size_t WordSize,
    uint32 Attributes )
```

12.113.2.40 CFE_PSP_MemRead16()

```
int32 CFE_PSP_MemRead16 (
    cpuaddr MemoryAddress,
    uint16 * uint16Value )
```

12.113.2.41 CFE_PSP_MemRead32()

```
int32 CFE_PSP_MemRead32 (
    cpuaddr MemoryAddress,
    uint32 * uint32Value )
```

12.113.2.42 CFE_PSP_MemRead8()

```
int32 CFE_PSP_MemRead8 (
    cpuaddr MemoryAddress,
    uint8 * ByteValue )
```

12.113.2.43 CFE_PSP_MemSet()

```
int32 CFE_PSP_MemSet (
    void * dest,
    uint8 value,
    uint32 n )
```

12.113.2.44 CFE_PSP_MemValidateRange()

```
int32 CFE_PSP_MemValidateRange (
    cpuaddr Address,
    size_t Size,
    uint32 MemoryType )
```

12.113.2.45 CFE_PSP_MemWrite16()

```
int32 CFE_PSP_MemWrite16 (
    cpuaddr MemoryAddress,
    uint16 uint16Value )
```

12.113.2.46 CFE_PSP_MemWrite32()

```
int32 CFE_PSP_MemWrite32 (
    cpuaddr MemoryAddress,
    uint32 uint32Value )
```

12.113.2.47 CFE_PSP_MemWrite8()

```
int32 CFE_PSP_MemWrite8 (
    cpuaddr MemoryAddress,
    uint8 ByteValue )
```

12.113.2.48 CFE_PSP_Panic()

```
void CFE_PSP_Panic (
    int32 ErrorCode )
```

12.113.2.49 CFE_PSP_PortRead16()

```
int32 CFE_PSP_PortRead16 (
    cpuaddr PortAddress,
    uint16 * uint16Value )
```

12.113.2.50 CFE_PSP_PortRead32()

```
int32 CFE_PSP_PortRead32 (
    cpuaddr PortAddress,
    uint32 * uint32Value )
```

12.113.2.51 CFE_PSP_PortRead8()

```
int32 CFE_PSP_PortRead8 (
    cpuaddr PortAddress,
    uint8 * ByteValue )
```

12.113.2.52 CFE_PSP_PortWrite16()

```
int32 CFE_PSP_PortWrite16 (
    cpuaddr PortAddress,
    uint16 uint16Value )
```

12.113.2.53 CFE_PSP_PortWrite32()

```
int32 CFE_PSP_PortWrite32 (
    cpuaddr PortAddress,
    uint32 uint32Value )
```

12.113.2.54 CFE_PSP_PortWrite8()

```
int32 CFE_PSP_PortWrite8 (
    cpuaddr PortAddress,
    uint8 ByteValue )
```

12.113.2.55 CFE_PSP_ReadFromCDS()

```
int32 CFE_PSP_ReadFromCDS (
    void * PtrToDataToRead,
    uint32 CDSOffset,
    uint32 NumBytes )
```

12.113.2.56 CFE_PSP_Restart()

```
void CFE_PSP_Restart (
    uint32 resetType )
```

12.113.2.57 CFE_PSP_SetDefaultExceptionEnvironment()

```
void CFE_PSP_SetDefaultExceptionEnvironment (
    void )
```

12.113.2.58 CFE_PSP_WatchdogDisable()

```
void CFE_PSP_WatchdogDisable (
    void )
```

12.113.2.59 CFE_PSP_WatchdogEnable()

```
void CFE_PSP_WatchdogEnable (
    void )
```

Referenced by HS_AppMain().

12.113.2.60 CFE_PSP_WatchdogGet()

```
uint32 CFE_PSP_WatchdogGet (
    void )
```

12.113.2.61 CFE_PSP_WatchdogInit()

```
void CFE_PSP_WatchdogInit (
    void )
```

12.113.2.62 CFE_PSP_WatchdogService()

```
void CFE_PSP_WatchdogService (
    void )
```

Referenced by HS_AppMain(), and HS_ProcessMain().

12.113.2.63 CFE_PSP_WatchdogSet()

```
void CFE_PSP_WatchdogSet (
    uint32 WatchdogValue )
```

Referenced by HS_AppMain().

12.113.2.64 CFE_PSP_WriteToCDS()

```
int32 CFE_PSP_WriteToCDS (
    const void * PtrToDataToWrite,
    uint32 CDSOffset,
    uint32 NumBytes )
```


Index

EXTENSION
 common_types.h, 1444

__attribute__
 hs_amt.c, 988
 hs_emt.c, 989
 hs_mat.c, 990
 hs_xct.c, 991

AMTableHandle
 HS_AppData_t, 794

AMTablePtr
 HS_AppData_t, 794

ARGCHECK
 osapi-macros.h, 1472

accuracy
 OS_timebase_prop_t, 847
 OS_timer_prop_t, 849

ActionType
 HS_AMTEEntry_t, 791
 HS_EMTEEntry_t, 811

ActiveBuffer
 CFE_TBL_HousekeepingTlm_Payload, 733

ActiveBufferAddr
 CFE_TBL_TblRegPacket_Payload, 750

ActiveTableFlag
 CFE_TBL_DumpCmd_Payload, 726
 CFE_TBL_ValidateCmd_Payload, 756

ActualLength
 OS_SockAddr_t, 840

addr
 OS_module_prop_t, 836

AddrData
 OS_SockAddr_t, 840

Address
 OS_static_symbol_record_t, 843

AddressesAreValid
 CFE_ES_AppInfo, 587

AlignPtr
 OS_SockAddrData_t, 841

AlignU32
 OS_SockAddrData_t, 841

AlivenessCounter
 HS_AppData_t, 794

AppData
 CFE_EVS_HousekeepingTlm_Payload, 662

AppDataFilename
 CFE_EVS_AppDataCmd_Payload, 647

AppEnableStatus
 CFE_EVS_AppTlmData, 656

AppEntryPoint
 CFE_ES_StartAppCmd_Payload, 639

AppFileName
 CFE_ES_AppReloadCmd_Payload, 595
 CFE_ES_StartAppCmd_Payload, 639

AppID
 CFE_EVS_AppTlmData, 656

AppId
 CFE_ES_TaskInfo, 645
 CFE_SB_PipeInfoEntry, 699

AppInfo
 CFE_ES_OneAppTlm_Payload, 623

AppMessageSentCounter
 CFE_EVS_AppTlmData, 657

AppMonCheckInCountdown
 HS_AppData_t, 795

AppMonEnables
 HS_AppData_t, 795
 HS_HkPacket_t, 813

AppMonLastExeCount
 HS_AppData_t, 795

AppMonLoaded
 HS_AppData_t, 795

AppName
 CFE_ES_TaskInfo, 645
 CFE_EVS_AppNameBitMaskCmd_Payload, 649
 CFE_EVS_AppNameCmd_Payload, 651
 CFE_EVS_AppNameEventIDCmd_Payload, 653
 CFE_EVS_AppNameEventIDMaskCmd_Payload,
 655
 CFE_EVS_PacketID, 671
 CFE_SB_PipeInfoEntry, 699
 CFE_SB_RoutingFileEntry, 705
 HS_AMTEEntry_t, 791
 HS_EMTEEntry_t, 811

Application
 CFE_ES_AppNameCmd_Payload, 594
 CFE_ES_AppReloadCmd_Payload, 595
 CFE_ES_SendMemPoolStatsCmd_Payload, 631
 CFE_ES_StartAppCmd_Payload, 639

ApplicationID
 CFE_FS_Header, 682

apps/hs/docs/dox_src/cfs_hs.dox, 850

apps/hs/fsw/mission_inc/hs_perfids.h, 850

apps/hs/fsw/platform_inc/hs_msgids.h, 850

apps/hs/fsw/platform_inc/hs_platform_cfg.h, 851

apps/hs/fsw/src/hs_app.c, 853

apps/hs/fsw/src/hs_app.h, 865

apps/hs/fsw/src/hs_cmds.c, 878

apps/hs/fsw/src/hs_cmds.h, 898

apps/hs/fsw/src/hs_custom.c, 918

apps/hs/fsw/src/hs_custom.h, 929

apps/hs/fsw/src/hs_events.h, 942

apps/hs/fsw/src/hs_monitors.c, [945](#)
 apps/hs/fsw/src/hs_monitors.h, [954](#)
 apps/hs/fsw/src/hs_msg.h, [964](#)
 apps/hs/fsw/src/hs_msgdefs.h, [965](#)
 apps/hs/fsw/src/hs_tbl.h, [968](#)
 apps/hs/fsw/src/hs_tbldefs.h, [969](#)
 apps/hs/fsw/src/hs_utils.c, [980](#)
 apps/hs/fsw/src/hs_utils.h, [984](#)
 apps/hs/fsw/src/hs_verify.h, [987](#)
 apps/hs/fsw/src/hs_version.h, [987](#)
 apps/hs/fsw/tables/hs_amt.c, [988](#)
 apps/hs/fsw/tables/hs_emt.c, [989](#)
 apps/hs/fsw/tables/hs_mat.c, [990](#)
 apps/hs/fsw/tables/hs_xct.c, [991](#)

AtToneDelay
 CFE_TIME_DiagnosticTlm_Payload, [760](#)

AtToneLatch
 CFE_TIME_DiagnosticTlm_Payload, [760](#)

AtToneLeapSeconds
 CFE_TIME_DiagnosticTlm_Payload, [760](#)
 CFE_TIME_ToneDataCmd_Payload, [790](#)

AtToneMET
 CFE_TIME_DiagnosticTlm_Payload, [760](#)
 CFE_TIME_ToneDataCmd_Payload, [790](#)

AtToneSTCF
 CFE_TIME_DiagnosticTlm_Payload, [760](#)
 CFE_TIME_ToneDataCmd_Payload, [790](#)

AtToneState
 CFE_TIME_ToneDataCmd_Payload, [790](#)

BSSAddress
 CFE_ES_AppInfo, [587](#)

BSSSize
 CFE_ES_AppInfo, [588](#)

BUFF_SIZE
 cfe_psp.h, [1496](#)

BUGCHECK
 osapi-macros.h, [1472](#)

BUGREPORT
 osapi-macros.h, [1473](#)

BitMask
 CFE_EVS_AppNameBitMaskCmd_Payload, [649](#)
 CFE_EVS_BitMaskCmd_Payload, [660](#)

block_size
 OS_statvfs_t, [844](#)

BlockSize
 CFE_ES_BlockStats, [596](#)

BlockStats
 CFE_ES_MemPoolStats, [618](#)

blocks_free
 OS_statvfs_t, [844](#)

BootSource
 CFE_ES_HousekeepingTlm_Payload, [607](#)

bss_address

 OS_module_address_t, [834](#)

bss_size
 OS_module_address_t, [834](#)

Buffer
 HS_MATMsgBuf_t, [819](#)
 OS_SockAddrData_t, [841](#)

build/docs/osconfig-example.h, [992](#)

ByteAlign4
 CFE_TBL_TblRegPacket_Payload, [750](#)

ByteAlignPad1
 CFE_TBL_HousekeepingTlm_Payload, [733](#)

ByteAlignSpare
 CFE_ES_CDSRegDumpRec, [597](#)

CCSDS_ExtendedHeader, [584](#)
 Subsystem, [584](#)
 SystemId, [584](#)

CCSDS_ExtendedHeader_t
 ccsds_hdr.h, [1314](#)

CCSDS_PrimaryHeader, [585](#)
 Length, [585](#)
 Sequence, [585](#)
 StreamId, [585](#)

CCSDS_PrimaryHeader_t
 ccsds_hdr.h, [1314](#)

CDSData
 HS_AppData_t, [796](#)

CDSState
 HS_AppData_t, [796](#)

cFE Access Table Content APIs, [406](#)
 CFE_TBL_GetAddress, [406](#)
 CFE_TBL_GetAddresses, [407](#)
 CFE_TBL_ReleaseAddress, [408](#)
 CFE_TBL_ReleaseAddresses, [409](#)

cFE Application Behavior APIs, [269](#)
 CFE_ES_ExitApp, [269](#)
 CFE_ES_IncrementTaskCounter, [270](#)
 CFE_ES_RunLoop, [270](#)
 CFE_ES_WaitForStartupSync, [272](#)
 CFE_ES_WaitForSystemState, [272](#)

cFE Application Control APIs, [266](#)
 CFE_ES_DeleteApp, [266](#)
 CFE_ES_ReloadApp, [267](#)
 CFE_ES_RestartApp, [268](#)

cFE Child Task APIs, [284](#)
 CFE_ES_CreateChildTask, [284](#)
 CFE_ES_DeleteChildTask, [285](#)
 CFE_ES_ExitChildTask, [286](#)
 CFE_ES_GetTaskIDByName, [286](#)
 CFE_ES_GetTaskName, [287](#)

cFE Clock State Flag Defines, [440](#)
 CFE_TIME_FLAG_ADD1HZ, [440](#)
 CFE_TIME_FLAG_ADDADJ, [440](#)
 CFE_TIME_FLAG_ADDTCL, [441](#)

CFE_TIME_FLAG_CLKSET, 441
CFE_TIME_FLAG_CMDFLY, 441
CFE_TIME_FLAG_FLYING, 441
CFE_TIME_FLAG_GDTONE, 441
CFE_TIME_FLAG_REFERR, 442
CFE_TIME_FLAG_SERVER, 442
CFE_TIME_FLAG_SIGPRI, 442
CFE_TIME_FLAG_SRCINT, 442
CFE_TIME_FLAG_SRVFLY, 442
CFE_TIME_FLAG_UNUSED, 442

cFE Critical Data Store APIs, 293
CFE_ES_CopyToCDS, 293
CFE_ES_GetCDSBlockIDByName, 294
CFE_ES_GetCDSBlockName, 295
CFE_ES_RegisterCDS, 296
CFE_ES_RestoreFromCDS, 297

cFE Entry/Exit APIs, 264
CFE_ES_Main, 264
CFE_ES_ResetCFE, 265

cFE External Time Source APIs, 431
CFE_TIME_ExternalGPS, 431
CFE_TIME_ExternalMET, 432
CFE_TIME_ExternalTime, 432
CFE_TIME_ExternalTone, 433
CFE_TIME_RegisterSynchCallback, 434
CFE_TIME_UnregisterSynchCallback, 434

cFE File Header Management APIs, 326
CFE_FS_InitHeader, 326
CFE_FS_ReadHeader, 326
CFE_FS_SetTimestamp, 327
CFE_FS_WriteHeader, 328

cFE File Utility APIs, 331
CFE_FS_BackgroundFileDumpIsPending, 331
CFE_FS_BackgroundFileDumpRequest, 332
CFE_FS_ExtractFilenameFromPath, 332
CFE_FS_GetDefaultExtension, 333
CFE_FS_GetDefaultMountPoint, 334
CFE_FS_ParseInputFileName, 334
CFE_FS_ParseInputFileNameEx, 335

cFE Generic Counter APIs, 310
CFE_ES_DeleteGenCounter, 310
CFE_ES_GetGenCount, 311
CFE_ES_GetGenCounterIDByName, 312
CFE_ES_GetGenCounterName, 312
CFE_ES_IncrementGenCounter, 313
CFE_ES_RegisterGenCounter, 314
CFE_ES_SetGenCount, 315

cFE Generic Message APIs, 337
CFE_MSG_Init, 337

cFE Get Current Time APIs, 418
CFE_TIME_GetMETseconds, 418
CFE_TIME_GetMETsubsecs, 419
CFE_TIME_GetMET, 418
CFE_TIME_GetTAI, 419

CFE_TIME_GetTime, 420
CFE_TIME_GetUTC, 421

cFE Get Table Information APIs, 411
CFE_TBL_GetInfo, 411
CFE_TBL_GetStatus, 412
CFE_TBL_NotifyByMessage, 413

cFE Get Time Information APIs, 422
CFE_TIME_GetClockInfo, 422
CFE_TIME_GetClockState, 422
CFE_TIME_GetLeapSeconds, 423
CFE_TIME_GetSTCF, 423

cFE Information APIs, 274
CFE_ES_GetAppIDByName, 275
CFE_ES_GetAppID, 274
CFE_ES_GetAppInfo, 276
CFE_ES_GetAppName, 277
CFE_ES_GetLibIDByName, 277
CFE_ES_GetLibInfo, 278
CFE_ES_GetLibName, 279
CFE_ES_GetModuleInfo, 280
CFE_ES_GetResetType, 281
CFE_ES_GetTaskID, 282
CFE_ES_GetTaskInfo, 283

cFE Manage Table Content APIs, 399
CFE_TBL_DumpToBuffer, 399
CFE_TBL_Load, 400
CFE_TBL_Manage, 401
CFE_TBL_Modified, 402
CFE_TBL_Update, 403
CFE_TBL_Validate, 404

cFE Memory Manager APIs, 299
CFE_ES_GetMemPoolStats, 299
CFE_ES_GetPoolBuf, 300
CFE_ES_GetPoolBufInfo, 301
CFE_ES_PoolCreate, 302
CFE_ES_PoolCreateEx, 303
CFE_ES_PoolCreateNoSem, 304
CFE_ES_PoolDelete, 305
CFE_ES_PutPoolBuf, 306

cFE Message Characteristics APIs, 384
CFE_SB_GetUserData, 384
CFE_SB_GetUserDataLength, 385
CFE_SB_MessageStringGet, 385
CFE_SB_MessageStringSet, 386
CFE_SB_SetUserDataLength, 387
CFE_SB_TimeStampMsg, 388

cFE Message Extended Header APIs, 349
CFE_MSG_GetEDSVersion, 349
CFE_MSG_GetEndian, 350
CFE_MSG_GetPlaybackFlag, 350
CFE_MSG_GetSubsystem, 351
CFE_MSG_GetSystem, 352
CFE_MSG_SetEDSVersion, 352
CFE_MSG_SetEndian, 353

CFE_MSG_SetPlaybackFlag, 35
CFE_MSG_SetSubsystem, 354
CFE_MSG_SetSystem, 355

cFE Message ID APIs, [389](#)

[CFE_SB_IsValidMsgId](#), [389](#)
[CFE_SB_MsgId_Equal](#), [389](#)
[CFE_SB_MsgIdToValue](#), [390](#)
[CFE_SB_ValueToMsgId](#), [391](#)

cFE Message Id APIs, [362](#)

`CFE_MSG_GetMsgId`, 362
`CFE_MSG.GetTypeFromMsgId`, 363
`CFE_MSG_SetMsgId`, 363

cFE Message Primary Header APIs. 338

[CFE_MSG_GetApId](#), [338](#)
[CFE_MSG_GetHasSecondaryHeader](#), [339](#)
[CFE_MSG_GetHeaderVersion](#), [340](#)
[CFE_MSG_GetNextSequenceCount](#), [340](#)
[CFE_MSG_GetSegmentationFlag](#), [341](#)
[CFE_MSG_GetSequenceCount](#), [341](#)
[CFE_MSG.GetSize](#), [342](#)
[CFE_MSG.GetType](#), [343](#)
[CFE_MSG_SetApId](#), [343](#)
[CFE_MSG_SetHasSecondaryHeader](#), [345](#)
[CFE_MSG_SetHeaderVersion](#), [345](#)
[CFE_MSG_SetSegmentationFlag](#), [346](#)
[CFE_MSG_SetSequenceCount](#), [347](#)
[CFE_MSG_SetSize](#), [347](#)

CFE MSG_SetType, 348

Message Secondary Header APIs, 35
CFE_MSG_GenerateChecksum, 356
CFE_MSG_GetFcnCode, 357
CFE_MSG_GetMsgTime, 357
CFE_MSG_SetFcnCode, 358
CFE_MSG_SetMsgTime, 360
CFE_MSG_ValidateChecksum, 361

CFE_MSG_validateChecksum, 361

Message Subscription Control API
CFE_SB_Subscribe, 372
CFE_SB_SubscribeEx, 373
CFE_SB_SubscribeLocal, 374
CFE_SB_Unsubscribe, 375
CFE_SB_UnsubscribeLocal, 376

cEE Miscellaneous APIs 289

Miscellaneous API's, 289
CFE_ES_BackgroundWakeUp, 289
CFE_ES_CalculateCRC, 289
CFE_ES_ProcessAsyncEvent, 291
CFE_ES_WriteToSysLog, 291

cEE Miscellaneous Time APIs 436

CFE_TIME_Local1HzISR, 436
CFE_TIME_Print 436

cEE Performance Monitor APIs 307

[CFE_ES_PerfLogAdd](#), [308](#)
[CFE_ES_PerfLogEntry](#), [307](#)
[CFE_ES_PerfLogExit](#), [308](#)

CFE_ES_PerfLogExit, 308

[CFE_SB_CreatePipe](#), 365
[CFE_SB_DeletePipe](#), 366
[CFE_SB_GetPipeIdByName](#), 367
[CFE_SB_GetPipeName](#), 368
[CFE_SB_GetPipeOpts](#), 368
[CFE_SB_PipeId_ToIndex](#), 370
[CFE_SB_SetPipeOpts](#), 371

cFE Registration APIs, 317, 393

[CFE_EVS_Register](#), [317](#)
[CFE_TBL_Register](#), [393](#)
[CFE_TBL_Share](#), [396](#)
[CFE_TBL_Unregister](#), [397](#)

cFE Reset Event Filter APIs, [324](#)

`CFE_EVS_ResetAllFilters`, [324](#)
`CFE_EVS_ResetFilter`, [324](#)

cFE Resource ID APIs, [260](#)

[CFE_ES_AppID_ToIndex](#), [260](#)
[CFE_ES_CounterID_ToIndex](#), [261](#)
[CFE_ES_LibID_ToIndex](#), [261](#)
[CFE_ES_TaskID_ToIndex](#), [262](#)

cFE Resource ID base values, 438

cFE Return Code Defines, [223](#)

CFE_ES_APP_CLEANUP_ERR, 228
CFE_ES_BAD_ARGUMENT, 228
CFE_ES_BIN_SEM_DELETE_ERR, 228
CFE_ES_BUFFER_NOT_IN_POOL, 229
CFE_ES_CDS_ACCESS_ERROR, 229
CFE_ES_CDS_ALREADY_EXISTS, 229
CFE_ES_CDS_BLOCK_CRC_ERR, 229

CFE_ES_CDS_INSUFFICIENT_MEMORY, 230
CFE_ES_CDS_INVALID_NAME, 230
CFE_ES_CDS_INVALID_SIZE, 230
CFE_ES_CDS_INVALID, 230
CFE_ES_CDS_OWNER_ACTIVE_ERR, 231
CFE_ES_CDS_WRONG_TYPE_ERR, 231
CFE_ES_COUNT_SEM_DELETE_ERR, 231
CFE_ES_ERR_APP_CREATE, 231
CFE_ES_ERR_APP_REGISTER, 232
CFE_ES_ERR_CHILD_TASK_CREATE, 232
CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_T←
ASK 232

ASK, 232
CFE_ES_ERR_CHILD_TASK_DELETE, 232
CFE_ES_ERR_CHILD_TASK_REGISTER, 233
CFE_ES_ERR_DUPLICATE_NAME, 233
CFE_ES_ERR_LOAD_LIB, 233
CFE_ES_ERR_MEM_BLOCK_SIZE, 233
CFE_ES_ERR_NAME_NOT_FOUND, 234
CFE_ES_ERR_RESOURCEID_NOT_VALID, 234
CFE_ES_ERR_SYS_LOG_FULL, 234
CFE_ES_ERR_SYS_LOG_TRUNCATED, 234
CFE_ES_FILE_CLOSE_ERR, 235
CFE_ES_FILE_IO_ERR, 235
CFE_ES_LIB_ALREADY_LOADED, 235
CFE_ES_MUT_SEM_DELETE_ERR, 235

CFE_ES_NO_RESOURCE_IDS_AVAILABLE, 236
CFE_ES_NOT_IMPLEMENTED, 236
CFE_ES_OPERATION_TIMED_OUT, 236
CFE_ES_POOL_BLOCK_INVALID, 236
CFE_ES_QUEUE_DELETE_ERR, 237
CFE_ES_RST_ACCESS_ERR, 237
CFE_ES_TASK_DELETE_ERR, 237
CFE_ES_TIMER_DELETE_ERR, 237
CFE_EVS_APP_FILTER_OVERLOAD, 238
CFE_EVS_APP_ILLEGAL_APP_ID, 238
CFE_EVS_APP_NOT_REGISTERED, 238
CFE_EVS_EVT_NOT_REGISTERED, 238
CFE_EVS_FILE_WRITE_ERROR, 239
CFE_EVS_INVALID_PARAMETER, 239
CFE_EVS_NOT_IMPLEMENTED, 239
CFE_EVS_RESET_AREA_POINTER, 239
CFE_EVS_UNKNOWN_FILTER, 240
CFE_FS_BAD_ARGUMENT, 240
CFE_FS_FNAME_TOO_LONG, 240
CFE_FS_INVALID_PATH, 240
CFE_FS_NOT_IMPLEMENTED, 241
CFE_SB_BAD_ARGUMENT, 241
CFE_SB_BUF_ALOC_ERR, 241
CFE_SB_BUFFER_INVALID, 241
CFE_SB_INTERNAL_ERR, 242
CFE_SB_MAX_DESTS_MET, 242
CFE_SB_MAX_MSGS_MET, 242
CFE_SB_MAX_PIPES_MET, 242
CFE_SB_MSG_TOO_BIG, 243
CFE_SB_NO_MESSAGE, 243
CFE_SB_NOT_IMPLEMENTED, 243
CFE_SB_PIPE_CR_ERR, 243
CFE_SB_PIPE_RD_ERR, 244
CFE_SB_TIME_OUT, 244
CFE_SB_WRONG_MSG_TYPE, 244
CFE_STATUS_BAD_COMMAND_CODE, 244
CFE_STATUS_EXTERNAL_RESOURCE_FAIL, 245
CFE_STATUS_NO_COUNTER_INCREMENT, 245
CFE_STATUS_NOT_IMPLEMENTED, 245
CFE_STATUS_REQUEST_ALREADY_PENDING, 245
CFE_STATUS_UNKNOWN_MSG_ID, 246
CFE_STATUS_WRONG_MSG_LENGTH, 246
CFE_SUCCESS, 246
CFE_TBL_BAD_ARGUMENT, 246
CFE_TBL_ERR_ACCESS, 247
CFE_TBL_ERR_BAD_CONTENT_ID, 247
CFE_TBL_ERR_BAD_PROCESSOR_ID, 247
CFE_TBL_ERR_BAD_SPACECRAFT_ID, 247
CFE_TBL_ERR_BAD_SUBTYPE_ID, 248
CFE_TBL_ERR_DUMP_ONLY, 248
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE, 248
CFE_TBL_ERR_DUPLICATE_NOT_OWNED, 248
CFE_TBL_ERR_FILE_FOR_WRONG_TABLE, 249
CFE_TBL_ERR_FILE_SIZE_INCONSISTENT, 249
CFE_TBL_ERR_FILE_TOO_LARGE, 249
CFE_TBL_ERR_FILENAME_TOO_LONG, 249
CFE_TBL_ERR_HANDLES_FULL, 250
CFE_TBL_ERR_ILLEGAL_SRC_TYPE, 250
CFE_TBL_ERR_INVALID_HANDLE, 250
CFE_TBL_ERR_INVALID_NAME, 250
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, 253
CFE_TBL_ERR_NO_TBL_HEADER, 253
CFE_TBL_ERR_PARTIAL_LOAD, 253
CFE_TBL_ERR_REGISTRY_FULL, 253
CFE_TBL_ERR_SHORT_FILE, 254
CFE_TBL_ERR_UNREGISTERED, 254
CFE_TBL_INFO_DUMP_PENDING, 254
CFE_TBL_INFO_NO_UPDATE_PENDING, 254
CFE_TBL_INFO_NO_VALIDATION_PENDING, 255
CFE_TBL_INFO_RECOVERED_TBL, 255
CFE_TBL_INFO_TABLE_LOCKED, 255
CFE_TBL_INFO_UPDATE_PENDING, 255
CFE_TBL_INFO_UPDATED, 256
CFE_TBL_INFO_VALIDATION_PENDING, 256
CFE_TBL_MESSAGE_ERROR, 256
CFE_TBL_NOT_IMPLEMENTED, 256
CFE_TBL_WARN_DUPLICATE, 257
CFE_TBL_WARN_NOT_CRITICAL, 257
CFE_TBL_WARN_PARTIAL_LOAD, 257
CFE_TBL_WARN_SHORT_FILE, 257
CFE_TIME_BAD_ARGUMENT, 258
CFE_TIME_CALLBACK_NOT_REGISTERED, 258
CFE_TIME_INTERNAL_ONLY, 258
CFE_TIME_NOT_IMPLEMENTED, 258
CFE_TIME_OUT_OF_RANGE, 259
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS, 259
cFE SB Pipe options, 392
CFE_SB PIPEOPTS IGNOREMINE, 392
cFE Send Event APIs, 319
CFE_EVS_SendEvent, 319
CFE_EVS_SendEventWithAppID, 320
CFE_EVS_SendTimedEvent, 322
cFE Send/Receive Message APIs, 378
CFE_SB_ReceiveBuffer, 378
CFE_SB_TransmitMsg, 379
cFE Table Type Defines, 415
CFE_TBL_OPT_BUFFER_MSK, 415
CFE_TBL_OPT_CRITICAL_MSK, 416
CFE_TBL_OPT_CRITICAL, 415

CFE_TBL_OPT_DBL_BUFFER, 416
 CFE_TBL_OPT_DEFAULT, 416
 CFE_TBL_OPT_DUMP_ONLY, 416
 CFE_TBL_OPT_LD_DMP_MSK, 416
 CFE_TBL_OPT_LOAD_DUMP, 417
 CFE_TBL_OPT_NOT_CRITICAL, 417
 CFE_TBL_OPT_NOT_USR_DEF, 417
 CFE_TBL_OPT_SNGL_BUFFER, 417
 CFE_TBL_OPT_USR_DEF_ADDR, 417
 CFE_TBL_OPT_USR_DEF_MSK, 417
 cFE Time Arithmetic APIs, 425
 CFE_TIME_Add, 425
 CFE_TIME_Compare, 426
 CFE_TIME_Subtract, 427
 cFE Time Conversion APIs, 428
 CFE_TIME_MET2SCTime, 428
 CFE_TIME_Micro2SubSecs, 428
 CFE_TIME_Sub2MicroSecs, 429
 cFE Zero Copy APIs, 381
 CFE_SB_AllocateMessageBuffer, 381
 CFE_SB_ReleaseMessageBuffer, 382
 CFE_SB_TransmitBuffer, 382
 CFE_BIT
 cfe_sb.h, 1167
 CFE_BUILD_BASELINE
 cfe_version.h, 1196
 CFE_BUILD_NUMBER
 cfe_version.h, 1196
 CFE_CLR
 cfe_sb.h, 1168
 CFE_CONFIGID_UNDEFINED
 cfe_config_api_typedefs.h, 1097
 CFE_CONFIGID_C
 cfe_config_api_typedefs.h, 1097
 CFE_Config_Callback_t
 cfe_config_api_typedefs.h, 1097
 CFE_Config_GetIdByName
 cfe_config.h, 1092
 CFE_Config_GetName
 cfe_config.h, 1093
 CFE_Config_GetObjPointer
 cfe_config.h, 1093
 CFE_Config_GetString
 cfe_config.h, 1094
 CFE_Config_GetValue
 cfe_config.h, 1094
 CFE_Config_IterateAll
 cfe_config.h, 1096
 CFE_ConfigId_t
 cfe_config_api_typedefs.h, 1097
 CFE_ES_ALL_APPS_EID
 cfe_es_events.h, 1201
 CFE_ES_APP_CLEANUP_ERR
 cFE Return Code Defines, 228
 CFE_ES_APP_RESTART
 cfe_es_api_typedefs.h, 1113
 CFE_ES_APP_TLM_MID
 cpu1_msgids.h, 1002
 CFE_ES_APPID_UNDEFINED
 cfe_es_api_typedefs.h, 1113
 CFE_ES_APPID_C
 cfe_es_api_typedefs.h, 1113
 CFE_ES_AppID_ToIndex
 cFE Resource ID APIs, 260
 CFE_ES_AppId_t
 cfe_es_extern_typedefs.h, 1121
 CFE_ES_AppInfo, 586
 AddressesAreValid, 587
 BSSAddress, 587
 BSSSize, 588
 CodeAddress, 588
 CodeSize, 588
 DataAddress, 588
 DataSize, 589
 EntryPoint, 589
 ExceptionAction, 589
 ExecutionCounter, 589
 FileName, 590
 MainTaskId, 590
 MainTaskName, 590
 Name, 590
 NumOfChildTasks, 591
 Priority, 591
 Resourceld, 591
 StackSize, 591
 StartAddress, 592
 Type, 592
 CFE_ES_AppInfo_t
 cfe_es_extern_typedefs.h, 1121
 CFE_ES_AppNameCmd, 592
 CommandHeader, 593
 Payload, 593
 CFE_ES_AppNameCmd_Payload, 593
 Application, 594
 CFE_ES_AppNameCmd_Payload_t
 cfe_es_msg.h, 1259
 CFE_ES_AppNameCmd_t
 cfe_es_msg.h, 1259
 CFE_ES_AppReloadCmd_Payload, 594
 AppFileName, 595
 Application, 595
 CFE_ES_AppReloadCmd_Payload_t
 cfe_es_msg.h, 1259
 CFE_ES_AppState
 cfe_es_extern_typedefs.h, 1127
 CFE_ES_AppState_Enum_t
 cfe_es_extern_typedefs.h, 1121
 CFE_ES_AppType

cfe_es_extern_typedefs.h, 1127
CFE_ES_AppType_Enum_t
 cfe_es_extern_typedefs.h, 1121
CFE_ES_BAD_ARGUMENT
 cFE Return Code Defines, 228
CFE_ES_BIN_SEM_DELETE_ERR
 cFE Return Code Defines, 228
CFE_ES_BOOT_ERR_EID
 cfe_es_events.h, 1201
CFE_ES_BUFFER_NOT_IN_POOL
 cFE Return Code Defines, 229
CFE_ES_BUILD_INF_EID
 cfe_es_events.h, 1202
CFE_ES_BackgroundWakeups
 cFE Miscellaneous APIs, 289
CFE_ES_BlockStats, 595
 BlockSize, 596
 NumCreated, 596
 NumFree, 596
CFE_ES_BlockStats_t
 cfe_es_extern_typedefs.h, 1122
CFE_ES_CC1_ERR_EID
 cfe_es_events.h, 1202
CFE_ES_CDS_ACCESS_ERROR
 cFE Return Code Defines, 229
CFE_ES_CDS_ALREADY_EXISTS
 cFE Return Code Defines, 229
CFE_ES_CDS_BAD_HANDLE
 cfe_es_api_typedefs.h, 1113
CFE_ES_CDS_BLOCK_CRC_ERR
 cFE Return Code Defines, 229
CFE_ES_CDS_DELETE_ERR_EID
 cfe_es_events.h, 1203
CFE_ES_CDS_DELETE_TBL_ERR_EID
 cfe_es_events.h, 1203
CFE_ES_CDS_DELETED_INFO_EID
 cfe_es_events.h, 1203
CFE_ES_CDS_DUMP_ERR_EID
 cfe_es_events.h, 1204
CFE_ES_CDS_INSUFFICIENT_MEMORY
 cFE Return Code Defines, 230
CFE_ES_CDS_INVALID_NAME
 cFE Return Code Defines, 230
CFE_ES_CDS_INVALID_SIZE
 cFE Return Code Defines, 230
CFE_ES_CDS_INVALID
 cFE Return Code Defines, 230
CFE_ES_CDS_NAME_ERR_EID
 cfe_es_events.h, 1204
CFE_ES_CDS_OWNER_ACTIVE_EID
 cfe_es_events.h, 1204
CFE_ES_CDS_OWNER_ACTIVE_ERR
 cFE Return Code Defines, 231
CFE_ES_CDS_REG_DUMP_INF_EID
 cfe_es_events.h, 1205
CFE_ES_CDS_REGISTER_ERR_EID
 cfe_es_events.h, 1205
CFE_ES_CDS_WRONG_TYPE_ERR
 cFE Return Code Defines, 231
CFE_ES_CDSHANDLE_C
 cfe_es_api_typedefs.h, 1113
CFE_ES_CDCHandle_t
 cfe_es_extern_typedefs.h, 1122
CFE_ES_CDSRegDumpRec, 596
 ByteAlignSpare, 597
 Handle, 597
 Name, 597
 Size, 598
 Table, 598
CFE_ES_CDSRegDumpRec_t
 cfe_es_extern_typedefs.h, 1122
CFE_ES_CLEAR_ER_LOG_CC
 cfe_es_msg.h, 1235
CFE_ES_CLEAR_SYSLOG_CC
 cfe_es_msg.h, 1235
CFE_ES_CMD_MID
 cpu1_msgids.h, 1002
CFE_ES_COUNT_SEM_DELETE_ERR
 cFE Return Code Defines, 231
CFE_ES_COUNTERID_UNDEFINED
 cfe_es_api_typedefs.h, 1114
CFE_ES_COUNTERID_C
 cfe_es_api_typedefs.h, 1114
CFE_ES_CREATING_CDS_DUMP_ERR_EID
 cfe_es_events.h, 1205
CFE_ES_CalculateCRC
 cFE Miscellaneous APIs, 289
CFE_ES_ChildTaskMainFuncPtr_t
 cfe_es_api_typedefs.h, 1116
CFE_ES_ClearERLogCmd_t
 cfe_es_msg.h, 1259
CFE_ES_ClearSysLogCmd_t
 cfe_es_msg.h, 1259
CFE_ES_CopyToCDS
 cFE Critical Data Store APIs, 293
CFE_ES_CounterID_ToIndex
 cFE Resource ID APIs, 261
CFE_ES_CounterId_t
 cfe_es_extern_typedefs.h, 1122
CFE_ES_CreateChildTask
 cFE Child Task APIs, 284
CFE_ES_DBIT
 cfe_es.h, 1110
CFE_ES_DELETE_CDS_CC
 cfe_es_msg.h, 1236
CFE_ES_DTEST
 cfe_es.h, 1110
CFE_ES_DUMP_CDS_REGISTRY_CC

cfe_es_msg.h, [1237](#)
CFE_ES_DeleteApp
 cFE Application Control APIs, [266](#)
CFE_ES_DeleteCDSCmd, [598](#)
 CommandHeader, [599](#)
 Payload, [599](#)
CFE_ES_DeleteCDSCmd_Payload, [599](#)
 CdsName, [600](#)
CFE_ES_DeleteCDSCmd_Payload_t
 cfe_es_msg.h, [1259](#)
CFE_ES_DeleteCDSCmd_t
 cfe_es_msg.h, [1259](#)
CFE_ES_DeleteChildTask
 cFE Child Task APIs, [285](#)
CFE_ES_DeleteGenCounter
 cFE Generic Counter APIs, [310](#)
CFE_ES_DumpCDSRegistryCmd, [600](#)
 CommandHeader, [601](#)
 Payload, [601](#)
CFE_ES_DumpCDSRegistryCmd_Payload, [601](#)
 DumpFilename, [602](#)
CFE_ES_DumpCDSRegistryCmd_Payload_t
 cfe_es_msg.h, [1260](#)
CFE_ES_DumpCDSRegistryCmd_t
 cfe_es_msg.h, [1260](#)
CFE_ES_ERLOG1_INF_EID
 cfe_es_events.h, [1206](#)
CFE_ES_ERLOG2_EID
 cfe_es_events.h, [1206](#)
CFE_ES_ERLOG2_ERR_EID
 cfe_es_events.h, [1206](#)
CFE_ES_ERLOG_PENDING_ERR_EID
 cfe_es_events.h, [1207](#)
CFE_ES_ERR_APP_CREATE
 cFE Return Code Defines, [231](#)
CFE_ES_ERR_APP_REGISTER
 cFE Return Code Defines, [232](#)
CFE_ES_ERR_CHILD_TASK_CREATE
 cFE Return Code Defines, [232](#)
CFE_ES_ERR_CHILD_TASK_DELETE
 cFE Return Code Defines, [232](#)
CFE_ES_ERR_CHILD_TASK_REGISTER
 cFE Return Code Defines, [233](#)
CFE_ES_ERR_DUPLICATE_NAME
 cFE Return Code Defines, [233](#)
CFE_ES_ERR_LOAD_LIB
 cFE Return Code Defines, [233](#)
CFE_ES_ERR_MEM_BLOCK_SIZE
 cFE Return Code Defines, [233](#)
CFE_ES_ERR_NAME_NOT_FOUND
 cFE Return Code Defines, [234](#)
CFE_ES_ERR_RESOURCEID_NOT_VALID
 cFE Return Code Defines, [234](#)
CFE_ES_ERR_SYS_LOG_FULL
 cFE Return Code Defines, [234](#)
CFE_ES_ERR_SYS_LOG_TRUNCATED
 cFE Return Code Defines, [234](#)
CFE_ES_ERR_SYSLOGMODE_EID
 cfe_es_events.h, [1207](#)
CFE_ES_ERREXIT_APP_ERR_EID
 cfe_es_events.h, [1207](#)
CFE_ES_ERREXIT_APP_INF_EID
 cfe_es_events.h, [1208](#)
CFE_ES_EXIT_APP_ERR_EID
 cfe_es_events.h, [1208](#)
CFE_ES_EXIT_APP_INF_EID
 cfe_es_events.h, [1209](#)
CFE_ES_ExceptionAction
 cfe_es_extern_typedefs.h, [1128](#)
CFE_ES_ExceptionAction_Enum_t
 cfe_es_extern_typedefs.h, [1123](#)
CFE_ES_ExitApp
 cFE Application Behavior APIs, [269](#)
CFE_ES_ExitChildTask
 cFE Child Task APIs, [286](#)
CFE_ES_FILE_CLOSE_ERR
 cFE Return Code Defines, [235](#)
CFE_ES_FILE_IO_ERR
 cFE Return Code Defines, [235](#)
CFE_ES_FILEWRITE_ERR_EID
 cfe_es_events.h, [1209](#)
CFE_ES_FileNameCmd, [602](#)
 CommandHeader, [602](#)
 Payload, [602](#)
CFE_ES_FileNameCmd_Payload, [603](#)
 FileName, [603](#)
CFE_ES_FileNameCmd_Payload_t
 cfe_es_msg.h, [1260](#)
CFE_ES_FileNameCmd_t
 cfe_es_msg.h, [1260](#)
CFE_ES_GetAppIDByName
 cFE Information APIs, [275](#)
CFE_ES_GetAppID
 cFE Information APIs, [274](#)
CFE_ES_GetAppInfo
 cFE Information APIs, [276](#)
CFE_ES_GetAppName
 cFE Information APIs, [277](#)
CFE_ES_GetCDSBlockIDByName
 cFE Critical Data Store APIs, [294](#)
CFE_ES_GetCDSBlockName
 cFE Critical Data Store APIs, [295](#)
CFE_ES_GetGenCount
 cFE Generic Counter APIs, [311](#)
CFE_ES_GetGenCounterIDByName
 cFE Generic Counter APIs, [312](#)

CFE_ES_GetGenCounterName
 cFE Generic Counter APIs, 312

CFE_ES_GetLibIDByName
 cFE Information APIs, 277

CFE_ES_GetLibInfo
 cFE Information APIs, 278

CFE_ES_GetLibName
 cFE Information APIs, 279

CFE_ES_GetMemPoolStats
 cFE Memory Manager APIs, 299

CFE_ES_GetModuleInfo
 cFE Information APIs, 280

CFE_ES_GetPoolBuf
 cFE Memory Manager APIs, 300

CFE_ES_GetPoolBufInfo
 cFE Memory Manager APIs, 301

CFE_ES_GetResetType
 cFE Information APIs, 281

CFE_ES_GetTaskIDByName
 cFE Child Task APIs, 286

CFE_ES_GetTaskID
 cFE Information APIs, 282

CFE_ES_GetTaskInfo
 cFE Information APIs, 283

CFE_ES_GetTaskName
 cFE Child Task APIs, 287

CFE_ES_HK_TLM_MID
 cpu1_msgids.h, 1002

CFE_ES_HousekeepingTlm, 604
 Payload, 604
 TelemetryHeader, 604

CFE_ES_HousekeepingTlm_Payload, 604
 BootSource, 607
 CFECoreChecksum, 607
 CFEMajorVersion, 607
 CFEMinorVersion, 607
 CFEMissionRevision, 607
 CFERevision, 608
 CommandCounter, 608
 CommandErrorCounter, 608
 ERLogEntries, 608
 ERLogIndex, 609
 HeapBlocksFree, 609
 HeapBytesFree, 609
 HeapMaxBlockSize, 609
 MaxProcessorResets, 610
 OSALMajorVersion, 610
 OSALMinorVersion, 610
 OSALMissionRevision, 610
 OSALRevision, 611
 PSPMajorVersion, 613
 PSPMinorVersion, 614
 PSPMissionRevision, 614
 PSPRevision, 614

 PerfDataCount, 611
 PerfDataEnd, 611
 PerfDataStart, 611
 PerfDataToWrite, 612
 PerfFilterMask, 612
 PerfMode, 612
 PerfState, 612
 PerfTriggerCount, 613
 PerfTriggerMask, 613
 ProcessorResets, 613
 RegisteredCoreApps, 614
 RegisteredExternalApps, 615
 RegisteredLibs, 615
 RegisteredTasks, 615
 ResetSubtype, 615
 ResetType, 616
 SysLogBytesUsed, 616
 SysLogEntries, 616
 SysLogMode, 616
 SysLogSize, 617

 CFE_ES_HousekeepingTlm_Payload_t
 cfe_es_msg.h, 1260

 CFE_ES_HousekeepingTlm_t
 cfe_es_msg.h, 1260

 CFE_ES_INIT_INF_EID
 cfe_es_events.h, 1209

 CFE_ES_INITSTATS_INF_EID
 cfe_es_events.h, 1210

 CFE_ES_INVALID_POOL_HANDLE_ERR_EID
 cfe_es_events.h, 1210

 CFE_ES_IncrementGenCounter
 cFE Generic Counter APIs, 313

 CFE_ES_IncrementTaskCounter
 cFE Application Behavior APIs, 270

 CFE_ES_LEN_ERR_EID
 cfe_es_events.h, 1210

 CFE_ES_LIB_ALREADY_LOADED
 cFE Return Code Defines, 235

 CFE_ES_LIBID_UNDEFINED
 cfe_es_api_typedefs.h, 1114

 CFE_ES_LIBID_C
 cfe_es_api_typedefs.h, 1114

 CFE_ES_LibID_ToIndex
 cFE Resource ID APIs, 261

 CFE_ES_LibId_t
 cfe_es_extern_typedefs.h, 1123

 CFE_ES_LibraryEntryFuncPtr_t
 cfe_es_api_typedefs.h, 1117

 CFE_ES_LogEntryType
 cfe_es_extern_typedefs.h, 1128

 CFE_ES_LogEntryType_Enum_t
 cfe_es_extern_typedefs.h, 1123

 CFE_ES_LogMode
 cfe_es_extern_typedefs.h, 1128

CFE_ES_LogMode_Enum_t
 cfe_es_extern_typedefs.h, 1123

CFE_ES_MEMADDRESS_C
 cfe_es_extern_typedefs.h, 1120

CFE_ES_MEMHANDLE_UNDEFINED
 cfe_es_api_typedefs.h, 1114

CFE_ES_MEMHANDLE_C
 cfe_es_api_typedefs.h, 1114

CFE_ES_MEMOFFSET_C
 cfe_es_extern_typedefs.h, 1120

CFE_ES_MEMPOOLBUF_C
 cfe_es_api_typedefs.h, 1115

CFE_ES_MEMSTATS_TLM_MID
 cpu1_msgids.h, 1003

CFE_ES_MID_ERR_EID
 cfe_es_events.h, 1211

CFE_ES_MUT_SEM_DELETE_ERR
 cFE Return Code Defines, 235

CFE_ES_Main
 cFE Entry/Exit APIs, 264

CFE_ES_MemAddress_t
 cfe_es_extern_typedefs.h, 1124

CFE_ES_MemHandle_t
 cfe_es_extern_typedefs.h, 1124

CFE_ES_MemOffset_t
 cfe_es_extern_typedefs.h, 1124

CFE_ES_MemPoolBuf_t
 cfe_es_api_typedefs.h, 1117

CFE_ES_MemPoolStats, 617
 BlockStats, 618
 CheckErrCtr, 618
 NumBlocksRequested, 618
 NumFreeBytes, 619
 PoolSize, 619

CFE_ES_MemPoolStats_t
 cfe_es_extern_typedefs.h, 1125

CFE_ES_MemStatsTlm, 619
 Payload, 620
 TelemetryHeader, 620

CFE_ES_MemStatsTlm_t
 cfe_es_msg.h, 1261

CFE_ES_NO_MUTEX
 cfe_es_api_typedefs.h, 1115

CFE_ES_NO_RESOURCE_IDS_AVAILABLE
 cFE Return Code Defines, 236

CFE_ES_NOOP_CC
 cfe_es_msg.h, 1238

CFE_ES_NOOP_INF_EID
 cfe_es_events.h, 1211

CFE_ES_NOT_IMPLEMENTED
 cFE Return Code Defines, 236

CFE_ES_NoArgsCmd, 620
 CommandHeader, 621

CFE_ES_NoArgsCmd_t
 cfe_es_msg.h, 1231

CFE_ES_NoopCmd_t
 cfe_es_msg.h, 1261

CFE_ES_ONE_APP_EID
 cfe_es_events.h, 1211

CFE_ES_ONE_APPID_ERR_EID
 cfe_es_events.h, 1212

CFE_ES_ONE_ERR_EID
 cfe_es_events.h, 1212

CFE_ES_OPERATION_TIMED_OUT
 cFE Return Code Defines, 236

CFE_ES_OSCREATE_ERR_EID
 cfe_es_events.h, 1212

CFE_ES_OVER_WRITE_SYSLOG_CC
 cfe_es_msg.h, 1239

CFE_ES_OneAppTlm, 621
 Payload, 622
 TelemetryHeader, 622

CFE_ES_OneAppTlm_Payload, 622
 ApplInfo, 623

CFE_ES_OneAppTlm_Payload_t
 cfe_es_msg.h, 1261

CFE_ES_OneAppTlm_t
 cfe_es_msg.h, 1261

CFE_ES_OverWriteSysLogCmd, 623
 CommandHeader, 623
 Payload, 623

CFE_ES_OverWriteSysLogCmd_Payload, 624
 Mode, 624

CFE_ES_OverWriteSysLogCmd_Payload_t
 cfe_es_msg.h, 1261

CFE_ES_OverWriteSysLogCmd_t
 cfe_es_msg.h, 1262

CFE_ES_PCR_ERR1_EID
 cfe_es_events.h, 1213

CFE_ES_PCR_ERR2_EID
 cfe_es_events.h, 1213

CFE_ES_PERF_DATAWRITTEN_EID
 cfe_es_events.h, 1213

CFE_ES_PERF_FILTMSKCMD_EID
 cfe_es_events.h, 1214

CFE_ES_PERF_FILTMSKERR_EID
 cfe_es_events.h, 1214

CFE_ES_PERF_LOG_ERR_EID
 cfe_es_events.h, 1214

CFE_ES_PERF_STARTCMD_EID
 cfe_es_events.h, 1215

CFE_ES_PERF_STARTCMD_ERR_EID
 cfe_es_events.h, 1215

CFE_ES_PERF_STARTCMD_TRIG_ERR_EID
 cfe_es_events.h, 1215

CFE_ES_PERF_STOPCMD_EID
 cfe_es_events.h, 1216

CFE_ES_PERF_STOPCMD_ERR2_EID
 cfe_es_events.h, 1216

cfe_es_events.h, 1216
CFE_ES_PERF_TRIGMSKCMD_EID
 cfe_es_events.h, 1216
CFE_ES_PERF_TRIGMSKERR_EID
 cfe_es_events.h, 1217
CFE_ES_POOL_BLOCK_INVALID
 cFE Return Code Defines, 236
CFE_ES_PerfLogAdd
 cFE Performance Monitor APIs, 308
CFE_ES_PerfLogEntry
 cFE Performance Monitor APIs, 307
CFE_ES_PerfLogExit
 cFE Performance Monitor APIs, 308
CFE_ES_PoolAlign, 625
 LongDouble, 625
 LongInt, 625
 Ptr, 625
CFE_ES_PoolAlign_t
 cfe_es_api_typedefs.h, 1117
CFE_ES_PoolCreate
 cFE Memory Manager APIs, 302
CFE_ES_PoolCreateEx
 cFE Memory Manager APIs, 303
CFE_ES_PoolCreateNoSem
 cFE Memory Manager APIs, 304
CFE_ES_PoolDelete
 cFE Memory Manager APIs, 305
CFE_ES_PoolStatsTlm_Payload, 626
 PoolHandle, 626
 PoolStats, 626
CFE_ES_PoolStatsTlm_Payload_t
 cfe_es_msg.h, 1262
CFE_ES_ProcessAsyncEvent
 cFE Miscellaneous APIs, 291
CFE_ES_PutPoolBuf
 cFE Memory Manager APIs, 306
CFE_ES_QUERY_ALL_CC
 cfe_es_msg.h, 1240
CFE_ES_QUERY_ALL_TASKS_CC
 cfe_es_msg.h, 1241
CFE_ES_QUERY_ONE_CC
 cfe_es_msg.h, 1242
CFE_ES_QUEUE_DELETE_ERR
 cFE Return Code Defines, 237
CFE_ES_QueryAllCmd_t
 cfe_es_msg.h, 1262
CFE_ES_QueryAllTasksCmd_t
 cfe_es_msg.h, 1262
CFE_ES_QueryOneCmd_t
 cfe_es_msg.h, 1262
CFE_ES_RELOAD_APP_CC
 cfe_es_msg.h, 1243
CFE_ES_RELOAD_APP_DBG_EID
 cfe_es_events.h, 1217
CFE_ES_RELOAD_APP_ERR1_EID
 cfe_es_events.h, 1217
CFE_ES_RELOAD_APP_ERR2_EID
 cfe_es_events.h, 1218
CFE_ES_RELOAD_APP_ERR3_EID
 cfe_es_events.h, 1218
CFE_ES_RELOAD_APP_ERR4_EID
 cfe_es_events.h, 1218
CFE_ES_RELOAD_APP_INF_EID
 cfe_es_events.h, 1219
CFE_ES_RESET_COUNTERS_CC
 cfe_es_msg.h, 1244
CFE_ES_RESET_INF_EID
 cfe_es_events.h, 1219
CFE_ES_RESET_PR_COUNT_CC
 cfe_es_msg.h, 1245
CFE_ES_RESET_PR_COUNT_EID
 cfe_es_events.h, 1219
CFE_ES_RESTART_APP_CC
 cfe_es_msg.h, 1246
CFE_ES_RESTART_APP_DBG_EID
 cfe_es_events.h, 1220
CFE_ES_RESTART_APP_ERR1_EID
 cfe_es_events.h, 1220
CFE_ES_RESTART_APP_ERR2_EID
 cfe_es_events.h, 1220
CFE_ES_RESTART_APP_ERR3_EID
 cfe_es_events.h, 1221
CFE_ES_RESTART_APP_ERR4_EID
 cfe_es_events.h, 1221
CFE_ES_RESTART_APP_INF_EID
 cfe_es_events.h, 1221
CFE_ES_RESTART_CC
 cfe_es_msg.h, 1247
CFE_ES_RST_ACCESS_ERR
 cFE Return Code Defines, 237
CFE_ES_RegisterCDS
 cFE Critical Data Store APIs, 296
CFE_ES_RegisterGenCounter
 cFE Generic Counter APIs, 314
CFE_ES_ReloadApp
 cFE Application Control APIs, 267
CFE_ES_ReloadAppCmd, 627
 CommandHeader, 627
 Payload, 627
CFE_ES_ReloadAppCmd_t
 cfe_es_msg.h, 1262
CFE_ES_ResetCFE
 cFE Entry/Exit APIs, 265
CFE_ES_ResetCountersCmd_t
 cfe_es_msg.h, 1263
CFE_ES_ResetPRCountCmd_t
 cfe_es_msg.h, 1263
CFE_ES_RestartApp

cFE Application Control APIs, 268
CFE_ES_RestartAppCmd_t
 `cfe_es_msg.h`, 1263
CFE_ES_RestartCmd, 628
 CommandHeader, 628
 Payload, 628
CFE_ES_RestartCmd_Payload, 629
 RestartType, 629
CFE_ES_RestartCmd_Payload_t
 `cfe_es_msg.h`, 1263
CFE_ES_RestartCmd_t
 `cfe_es_msg.h`, 1263
CFE_ES_RestoreFromCDS
 cFE Critical Data Store APIs, 297
CFE_ES_RunLoop
 cFE Application Behavior APIs, 270
CFE_ES_RunStatus
 `cfe_es_extern_typedefs.h`, 1129
CFE_ES_RunStatus_Enum_t
 `cfe_es_extern_typedefs.h`, 1125
CFE_ES_SEND_HK_MID
 `cpu1_msghdr.h`, 1003
CFE_ES_SEND_MEM_POOL_STATS_CC
 `cfe_es_msg.h`, 1248
CFE_ES_SET_MAX_PR_COUNT_CC
 `cfe_es_msg.h`, 1249
CFE_ES_SET_MAX_PR_COUNT_EID
 `cfe_es_events.h`, 1222
CFE_ES_SET_PERF_FILTER_MASK_CC
 `cfe_es_msg.h`, 1250
CFE_ES_SET_PERF_TRIGGER_MASK_CC
 `cfe_es_msg.h`, 1251
CFE_ES_START_APP_CC
 `cfe_es_msg.h`, 1252
CFE_ES_START_ERR_EID
 `cfe_es_events.h`, 1222
CFE_ES_START_EXC_ACTION_ERR_EID
 `cfe_es_events.h`, 1222
CFE_ES_START_INF_EID
 `cfe_es_events.h`, 1223
CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID
 `cfe_es_events.h`, 1223
CFE_ES_START_INVALID_FILENAME_ERR_EID
 `cfe_es_events.h`, 1223
CFE_ES_START_NULL_APP_NAME_ERR_EID
 `cfe_es_events.h`, 1224
CFE_ES_START_PERF_DATA_CC
 `cfe_es_msg.h`, 1253
CFE_ES_START_PRIORITY_ERR_EID
 `cfe_es_events.h`, 1224
CFE_ES_STATIC_POOL_TYPE
 `cfe_es_api_typedefs.h`, 1115
CFE_ES_STOP_APP_CC
 `cfe_es_msg.h`, 1254
CFE_ES_STOP_DBG_EID
 `cfe_es_events.h`, 1224
CFE_ES_STOP_ERR1_EID
 `cfe_es_events.h`, 1225
CFE_ES_STOP_ERR2_EID
 `cfe_es_events.h`, 1225
CFE_ES_STOP_ERR3_EID
 `cfe_es_events.h`, 1225
CFE_ES_STOP_INF_EID
 `cfe_es_events.h`, 1226
CFE_ES_STOP_PERF_DATA_CC
 `cfe_es_msg.h`, 1255
CFE_ES_SYSLOG1_INF_EID
 `cfe_es_events.h`, 1226
CFE_ES_SYSLOG2_EID
 `cfe_es_events.h`, 1226
CFE_ES_SYSLOG2_ERR_EID
 `cfe_es_events.h`, 1227
CFE_ES_SYSLOGMODE_EID
 `cfe_es_events.h`, 1227
CFE_ES_SendMemPoolStatsCmd, 630
 CommandHeader, 630
 Payload, 630
CFE_ES_SendMemPoolStatsCmd_Payload, 631
 Application, 631
 PoolHandle, 631
CFE_ES_SendMemPoolStatsCmd_Payload_t
 `cfe_es_msg.h`, 1263
CFE_ES_SendMemPoolStatsCmd_t
 `cfe_es_msg.h`, 1264
CFE_ES_SetGenCount
 cFE Generic Counter APIs, 315
CFE_ES_SetMaxPRCountCmd, 632
 CommandHeader, 632
 Payload, 632
CFE_ES_SetMaxPRCountCmd_Payload, 633
 MaxPRCount, 633
CFE_ES_SetMaxPRCountCmd_Payload_t
 `cfe_es_msg.h`, 1264
CFE_ES_SetMaxPRCountCmd_t
 `cfe_es_msg.h`, 1264
CFE_ES_SetPerfFilterMaskCmd, 633
 CommandHeader, 634
 Payload, 634
CFE_ES_SetPerfFilterMaskCmd_Payload, 634
 FilterMask, 635
 FilterMaskNum, 635
CFE_ES_SetPerfFilterMaskCmd_Payload_t
 `cfe_es_msg.h`, 1264
CFE_ES_SetPerfFilterMaskCmd_t
 `cfe_es_msg.h`, 1264
CFE_ES_SetPerfTrigMaskCmd_Payload, 636
 TriggerMask, 637
 TriggerMaskNum, 637

CFE_ES_SetPerfTrigMaskCmd_Payload_t
 cfe_es_msg.h, 1265

CFE_ES_SetPerfTriggerMaskCmd, 635
 CommandHeader, 636
 Payload, 636

CFE_ES_SetPerfTriggerMaskCmd_t
 cfe_es_msg.h, 1264

CFE_ES_StackPointer_t
 cfe_es_api_typedefs.h, 1117

CFE_ES_StartApp, 637
 CommandHeader, 638
 Payload, 638

CFE_ES_StartAppCmd_Payload, 638
 AppEntryPoint, 639
 AppFileName, 639
 Application, 639
 ExceptionAction, 640
 Priority, 640
 StackSize, 640

CFE_ES_StartAppCmd_Payload_t
 cfe_es_msg.h, 1265

CFE_ES_StartAppCmd_t
 cfe_es_msg.h, 1265

CFE_ES_StartPerfCmd_Payload, 640
 TriggerMode, 641

CFE_ES_StartPerfCmd_Payload_t
 cfe_es_msg.h, 1265

CFE_ES_StartPerfDataCmd, 641
 CommandHeader, 642
 Payload, 642

CFE_ES_StartPerfDataCmd_t
 cfe_es_msg.h, 1265

CFE_ES_StopAppCmd_t
 cfe_es_msg.h, 1265

CFE_ES_StopPerfCmd_Payload, 642
 DataFileName, 643

CFE_ES_StopPerfCmd_Payload_t
 cfe_es_msg.h, 1266

CFE_ES_StopPerfDataCmd, 643
 CommandHeader, 644
 Payload, 644

CFE_ES_StopPerfDataCmd_t
 cfe_es_msg.h, 1266

CFE_ES_SystemState
 cfe_es_extern_typedefs.h, 1129

CFE_ES_SystemState_Enum_t
 cfe_es_extern_typedefs.h, 1125

CFE_ES_TASK_DELETE_ERR
 cFE Return Code Defines, 237

CFE_ES_TASK_STACK_ALLOCATE
 cfe_es_api_typedefs.h, 1115

CFE_ES_TASKID_UNDEFINED
 cfe_es_api_typedefs.h, 1116

CFE_ES_TASKID_C
 cfe_es_msg.h, 1266

 cfe_es_api_typedefs.h, 1116

CFE_ES_TASKINFO_EID
 cfe_es_events.h, 1227

CFE_ES_TASKINFO_OSCCREATE_ERR_EID
 cfe_es_events.h, 1228

CFE_ES_TASKINFO_WR_ERR_EID
 cfe_es_events.h, 1228

CFE_ES_TASKINFO_WRHDR_ERR_EID
 cfe_es_events.h, 1228

CFE_ES_TASKWR_ERR_EID
 cfe_es_events.h, 1229

CFE_ES_TEST_LONG_MASK
 cfe_es.h, 1111

CFE_ES_TIMER_DELETE_ERR
 cFE Return Code Defines, 237

CFE_ES_TLM_POOL_STATS_INFO_EID
 cfe_es_events.h, 1229

CFE_ES_TaskEntryFuncPtr_t
 cfe_es_api_typedefs.h, 1118

CFE_ES_TaskID_ToIndex
 cFE Resource ID APIs, 262

CFE_ES_TaskId_t
 cfe_es_extern_typedefs.h, 1126

CFE_ES_TaskInfo, 644
 Appld, 645
 AppName, 645
 ExecutionCounter, 645
 Priority, 645
 Spare, 646
 StackSize, 646
 TaskId, 646
 TaskName, 646

CFE_ES_TaskInfo_t
 cfe_es_extern_typedefs.h, 1126

CFE_ES_TaskPriority_Atom_t
 cfe_es_extern_typedefs.h, 1126

CFE_ES_USE_MUTEX
 cfe_es_api_typedefs.h, 1116

CFE_ES_VERSION_INF_EID
 cfe_es_events.h, 1229

CFE_ES_WRHDR_ERR_EID
 cfe_es_events.h, 1230

CFE_ES_WRITE_CFE_HDR_ERR_EID
 cfe_es_events.h, 1230

CFE_ES_WRITE_ER_LOG_CC
 cfe_es_msg.h, 1256

CFE_ES_WRITE_SYSLOG_CC
 cfe_es_msg.h, 1257

CFE_ES_WaitForStartupSync
 cFE Application Behavior APIs, 272

CFE_ES_WaitForSystemState
 cFE Application Behavior APIs, 272

CFE_ES_WriteERLogCmd_t
 cfe_es_msg.h, 1266

CFE_ES_WriteSysLogCmd_t
 cfe_es_msg.h, 1266

CFE_ES_WriteToSysLog
 cFE Miscellaneous APIs, 291

CFE_EVENTS_SERVICE
 cfe_error.h, 1105

CFE_EVS_ADD_EVENT_FILTER_CC
 cfe_evs_msg.h, 1285

CFE_EVS_ADDFILTER_EID
 cfe_evs_events.h, 1268

CFE_EVS_APP_FILTER_OVERLOAD
 cFE Return Code Defines, 238

CFE_EVS_APP_ILLEGAL_APP_ID
 cFE Return Code Defines, 238

CFE_EVS_APP_NOT_REGISTERED
 cFE Return Code Defines, 238

CFE_EVS_AddEventFilterCmd_t
 cfe_evs_msg.h, 1306

CFE_EVS_AppDataCmd_Payload, 647
 AppDataFilename, 647

CFE_EVS_AppDataCmd_Payload_t
 cfe_evs_msg.h, 1306

CFE_EVS_AppNameBitMaskCmd, 647
 CommandHeader, 648
 Payload, 648

CFE_EVS_AppNameBitMaskCmd_Payload, 648
 AppName, 649
 BitMask, 649
 Spare, 649

CFE_EVS_AppNameBitMaskCmd_Payload_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameBitMaskCmd_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameCmd, 650
 CommandHeader, 650
 Payload, 650

CFE_EVS_AppNameCmd_Payload, 651
 AppName, 651

CFE_EVS_AppNameCmd_Payload_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameCmd_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameEventIDCmd, 652
 CommandHeader, 652
 Payload, 652

CFE_EVS_AppNameEventIDCmd_Payload, 653
 AppName, 653
 EventID, 653

CFE_EVS_AppNameEventIDCmd_Payload_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameEventIDCmd_t
 cfe_evs_msg.h, 1307

CFE_EVS_AppNameEventIDMaskCmd, 654
 CommandHeader, 654

Payload, 654

CFE_EVS_AppNameEventIDMaskCmd_Payload, 655
 AppName, 655
 EventID, 655
 Mask, 655

CFE_EVS_AppNameEventIDMaskCmd_Payload_t
 cfe_evs_msg.h, 1308

CFE_EVS_AppNameEventIDMaskCmd_t
 cfe_evs_msg.h, 1308

CFE_EVS_AppTlmData, 656
 AppEnableStatus, 656
 AppID, 656
 AppMessageSentCounter, 657
 Padding, 657

CFE_EVS_AppTlmData_t
 cfe_evs_msg.h, 1308

CFE_EVS_BinFilter, 657
 EventID, 658
 Mask, 658

CFE_EVS_BinFilter_t
 cfe_evs_api_typedefs.h, 1135

CFE_EVS_BitMaskCmd, 658
 CommandHeader, 659
 Payload, 659

CFE_EVS_BitMaskCmd_Payload, 659
 BitMask, 660
 Spare, 660

CFE_EVS_BitMaskCmd_Payload_t
 cfe_evs_msg.h, 1308

CFE_EVS_BitMaskCmd_t
 cfe_evs_msg.h, 1308

CFE_EVS_CLEAR_LOG_CC
 cfe_evs_msg.h, 1285

CFE_EVS_CMD_MID
 cpu1_msgids.h, 1003

CFE_EVS_CRITICAL_BIT
 cfe_evs_msg.h, 1286

CFE_EVS_ClearLogCmd_t
 cfe_evs_msg.h, 1308

CFE_EVS_DEBUG_BIT
 cfe_evs_msg.h, 1286

CFE_EVS_DELETE_EVENT_FILTER_CC
 cfe_evs_msg.h, 1287

CFE_EVS_DELFILTER_EID
 cfe_evs_events.h, 1268

CFE_EVS_DISABLE_APP_EVENT_TYPE_CC
 cfe_evs_msg.h, 1287

CFE_EVS_DISABLE_APP_EVENTS_CC
 cfe_evs_msg.h, 1288

CFE_EVS_DISABLE_EVENT_TYPE_CC
 cfe_evs_msg.h, 1289

CFE_EVS_DISABLE_PORTS_CC
 cfe_evs_msg.h, 1290

CFE_EVS_DISAPPENTYPE_EID

cfe_evs_events.h, 1269
CFE_EVS_DISAPPEVT_EID
 cfe_evs_events.h, 1269
CFE_EVS_DISEVTTYPE_EID
 cfe_evs_events.h, 1269
CFE_EVS_DISPORT_EID
 cfe_evs_events.h, 1270
CFE_EVS_DeleteEventFilterCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_DisableAppEventTypeCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_DisableAppEventsCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_DisableEventTypeCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_DisablePortsCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_ENAAPPEVT_EID
 cfe_evs_events.h, 1270
CFE_EVS_ENAAPPEVTTYPE_EID
 cfe_evs_events.h, 1270
CFE_EVS_ENABLE_APP_EVENT_TYPE_CC
 cfe_evs_msg.h, 1291
CFE_EVS_ENABLE_APP_EVENTS_CC
 cfe_evs_msg.h, 1292
CFE_EVS_ENABLE_EVENT_TYPE_CC
 cfe_evs_msg.h, 1293
CFE_EVS_ENABLE_PORTS_CC
 cfe_evs_msg.h, 1294
CFE_EVS_ENAEVTTYPE_EID
 cfe_evs_events.h, 1271
CFE_EVS_ENAPORT_EID
 cfe_evs_events.h, 1271
CFE_EVS_ERR_APPNOREGS_EID
 cfe_evs_events.h, 1271
CFE_EVS_ERR_CC_EID
 cfe_evs_events.h, 1272
CFE_EVS_ERR_CRDATFILE_EID
 cfe_evs_events.h, 1272
CFE_EVS_ERR_CRLOGFILE_EID
 cfe_evs_events.h, 1272
CFE_EVS_ERR_EVTIDNOREGS_EID
 cfe_evs_events.h, 1273
CFE_EVS_ERR_ILLAPPIDRANGE_EID
 cfe_evs_events.h, 1273
CFE_EVS_ERR_ILLEGALFMTMOD_EID
 cfe_evs_events.h, 1273
CFE_EVS_ERR_INVALID_BITMASK_EID
 cfe_evs_events.h, 1274
CFE_EVS_ERR_LOGMODE_EID
 cfe_evs_events.h, 1274
CFE_EVS_ERR_MAXREGSFILTER_EID
 cfe_evs_events.h, 1274
CFE_EVS_ERR_MSGID_EID
 cfe_evs_events.h, 1275
CFE_EVS_ERR_NOAPPIDFOUND_EID
 cfe_evs_events.h, 1275
CFE_EVS_ERR_UNREGISTERED_EVS_APP
 cfe_evs_events.h, 1275
CFE_EVS_ERR_WRDATFILE_EID
 cfe_evs_events.h, 1276
CFE_EVS_ERR_WRLOGFILE_EID
 cfe_evs_events.h, 1276
CFE_EVS_ERROR_BIT
 cfe_evs_msg.h, 1295
CFE_EVS_EVERY_FOURTH_ONE
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_EVERY_OTHER_ONE
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_EVERY_OTHER_TWO
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_EVT_FILTERED_EID
 cfe_evs_events.h, 1276
CFE_EVS_EVT_NOT_REGISTERED
 cFE Return Code Defines, 238
CFE_EVS_EnableAppEventTypeCmd_t
 cfe_evs_msg.h, 1310
CFE_EVS_EnableAppEventsCmd_t
 cfe_evs_msg.h, 1309
CFE_EVS_EnableEventTypeCmd_t
 cfe_evs_msg.h, 1310
CFE_EVS_EnablePortsCmd_t
 cfe_evs_msg.h, 1310
CFE_EVS_EventFilter
 cfe_evs_extern_typedefs.h, 1137
CFE_EVS_EventFilter_Enum_t
 cfe_evs_extern_typedefs.h, 1136
CFE_EVS_EventOutput
 cfe_evs_extern_typedefs.h, 1137
CFE_EVS_EventOutput_Enum_t
 cfe_evs_extern_typedefs.h, 1136
CFE_EVS_EventType
 cfe_evs_extern_typedefs.h, 1139
CFE_EVS_EventType_Enum_t
 cfe_evs_extern_typedefs.h, 1136
CFE_EVS_FILE_WRITE_ERROR
 cFE Return Code Defines, 239
CFE_EVS_FILTER_MAX_EID
 cfe_evs_events.h, 1277
CFE_EVS_FIRST_16_STOP
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_FIRST_32_STOP
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_FIRST_4_STOP
 cfe_evs_api_typedefs.h, 1133
CFE_EVS_FIRST_64_STOP
 cfe_evs_api_typedefs.h, 1134
CFE_EVS_FIRST_8_STOP

cfe_evs_api_typedefs.h, 1134
 CFE_EVS_FIRST_ONE_STOP
 cfe_evs_api_typedefs.h, 1134
 CFE_EVS_FIRST_TWO_STOP
 cfe_evs_api_typedefs.h, 1134
 CFE_EVS_HK_TLM_MID
 cpu1_msgids.h, 1003
 CFE_EVS_HousekeepingTlm, 660
 Payload, 661
 TelemetryHeader, 661
 CFE_EVS_HousekeepingTlm_Payload, 661
 AppData, 662
 CommandCounter, 663
 CommandErrorCounter, 663
 LogEnabled, 663
 LogFullFlag, 663
 LogMode, 664
 LogOverflowCounter, 664
 MessageFormatMode, 664
 MessageSendCounter, 664
 MessageTruncCounter, 665
 OutputPort, 665
 Spare1, 665
 Spare2, 665
 Spare3, 666
 UnregisteredAppCounter, 666
 CFE_EVS_HousekeepingTlm_Payload_t
 cfe_evs_msg.h, 1310
 CFE_EVS_HousekeepingTlm_t
 cfe_evs_msg.h, 1310
 CFE_EVS_INFORMATION_BIT
 cfe_evs_msg.h, 1296
 CFE_EVS_INVALID_PARAMETER
 cFE Return Code Defines, 239
 CFE_EVS_LEN_ERR_EID
 cfe_evs_events.h, 1277
 CFE_EVS_LOGMODE_EID
 cfe_evs_events.h, 1277
 CFE_EVS_LONG_EVENT_MSG_MID
 cpu1_msgids.h, 1003
 CFE_EVS_LogFileCmd_Payload, 666
 LogFilename, 667
 CFE_EVS_LogFileCmd_Payload_t
 cfe_evs_msg.h, 1310
 CFE_EVS_LogMode
 cfe_evs_extern_typedefs.h, 1139
 CFE_EVS_LogMode_Enum_t
 cfe_evs_extern_typedefs.h, 1136
 CFE_EVS_LongEventTlm, 667
 Payload, 668
 TelemetryHeader, 668
 CFE_EVS_LongEventTlm_Payload, 668
 Message, 669
 PacketID, 669
 Spare1, 669
 Spare2, 669
 CFE_EVS_LongEventTlm_Payload_t
 cfe_evs_msg.h, 1311
 CFE_EVS_LongEventTlm_t
 cfe_evs_msg.h, 1311
 CFE_EVS_MsgFormat
 cfe_evs_extern_typedefs.h, 1139
 CFE_EVS_MsgFormat_Enum_t
 cfe_evs_extern_typedefs.h, 1137
 CFE_EVS_NO_FILTER
 cfe_evs_api_typedefs.h, 1134
 CFE_EVS_NOOP_CC
 cfe_evs_msg.h, 1296
 CFE_EVS_NOOP_EID
 cfe_evs_events.h, 1278
 CFE_EVS_NOT_IMPLEMENTED
 cFE Return Code Defines, 239
 CFE_EVS_NoArgsCmd, 670
 CommandHeader, 670
 CFE_EVS_NoArgsCmd_t
 cfe_evs_msg.h, 1311
 CFE_EVS_NoopCmd_t
 cfe_evs_msg.h, 1311
 CFE_EVS_PORT1_BIT
 cfe_evs_msg.h, 1296
 CFE_EVS_PORT2_BIT
 cfe_evs_msg.h, 1297
 CFE_EVS_PORT3_BIT
 cfe_evs_msg.h, 1297
 CFE_EVS_PORT4_BIT
 cfe_evs_msg.h, 1297
 CFE_EVS_PacketID_t
 cfe_evs_msg.h, 1311
 CFE_EVS_PacketID, 670
 AppName, 671
 EventID, 671
 EventType, 671
 ProcessorID, 671
 SpacecraftID, 672
 CFE_EVS_RESET_ALL_FILTERS_CC
 cfe_evs_msg.h, 1297
 CFE_EVS_RESET_APP_COUNTER_CC
 cfe_evs_msg.h, 1298
 CFE_EVS_RESET_AREA_POINTER
 cFE Return Code Defines, 239
 CFE_EVS_RESET_COUNTERS_CC
 cfe_evs_msg.h, 1299
 CFE_EVS_RESET_FILTER_CC
 cfe_evs_msg.h, 1300
 CFE_EVS_RSTALLFILTER_EID
 cfe_evs_events.h, 1278
 CFE_EVS_RSTCNT_EID
 cfe_evs_events.h, 1278

CFE_EVS_RSTEVTCNT_EID
 cfe_evs_events.h, 1279

CFE_EVS_RSTFILTER_EID
 cfe_evs_events.h, 1279

CFE_EVS_Register
 cFE Registration APIs, 317

CFE_EVS_ResetAllFilters
 cFE Reset Event Filter APIs, 324

CFE_EVS_ResetAllFiltersCmd_t
 cfe_evs_msg.h, 1311

CFE_EVS_ResetAppCounterCmd_t
 cfe_evs_msg.h, 1311

CFE_EVS_ResetCountersCmd_t
 cfe_evs_msg.h, 1312

CFE_EVS_ResetFilter
 cFE Reset Event Filter APIs, 324

CFE_EVS_ResetFilterCmd_t
 cfe_evs_msg.h, 1312

CFE_EVS_SEND_HK_MID
 cpu1_msgids.h, 1003

CFE_EVS_SET_EVENT_FORMAT_MODE_CC
 cfe_evs_msg.h, 1301

CFE_EVS_SET_FILTER_CC
 cfe_evs_msg.h, 1302

CFE_EVS_SET_LOG_MODE_CC
 cfe_evs_msg.h, 1303

CFE_EVS_SETEVTFMTMOD_EID
 cfe_evs_events.h, 1279

CFE_EVS_SETFILTERMSK_EID
 cfe_evs_events.h, 1280

CFE_EVS_SHORT_EVENT_MSG_MID
 cpu1_msgids.h, 1004

CFE_EVS_STARTUP_EID
 cfe_evs_events.h, 1280

CFE_EVS_Send
 cfe_evs.h, 1130

CFE_EVS_SendCrit
 cfe_evs.h, 1131

CFE_EVS_SendDbg
 cfe_evs.h, 1131

CFE_EVS_SendErr
 cfe_evs.h, 1131

CFE_EVS_SendEvent
 cFE Send Event APIs, 319

CFE_EVS_SendEventWithAppID
 cFE Send Event APIs, 320

CFE_EVS_SendInfo
 cfe_evs.h, 1131

CFE_EVS_SendTimedEvent
 cFE Send Event APIs, 322

CFE_EVS_SetEventFormatCode_Payload, 672
 MsgFormat, 673
 Spare, 673

CFE_EVS_SetEventFormatMode_Payload_t
 cfe_evs_msg.h, 1312

CFE_EVS_SetEventFormatModeCmd, 673
 CommandHeader, 674
 Payload, 674

CFE_EVS_SetEventFormatModeCmd_t
 cfe_evs_msg.h, 1312

CFE_EVS_SetFilterCmd_t
 cfe_evs_msg.h, 1312

CFE_EVS_SetLogMode_Payload, 674
 LogMode, 675
 Spare, 675

CFE_EVS_SetLogMode_Payload_t
 cfe_evs_msg.h, 1312

CFE_EVS_SetLogModeCmd, 675
 CommandHeader, 676
 Payload, 676

CFE_EVS_SetLogModeCmd_t
 cfe_evs_msg.h, 1313

CFE_EVS_ShortEventTlm, 676
 Payload, 676
 TelemetryHeader, 677

CFE_EVS_ShortEventTlm_Payload, 677
 PacketID, 677

CFE_EVS_ShortEventTlm_Payload_t
 cfe_evs_msg.h, 1313

CFE_EVS_ShortEventTlm_t
 cfe_evs_msg.h, 1313

CFE_EVS_UNKNOWN_FILTER
 cFE Return Code Defines, 240

CFE_EVS_WRDAT_EID
 cfe_evs_events.h, 1280

CFE_EVS_WRITE_APP_DATA_FILE_CC
 cfe_evs_msg.h, 1304

CFE_EVS_WRITE_LOG_DATA_FILE_CC
 cfe_evs_msg.h, 1305

CFE_EVS_WRLOG_EID
 cfe_evs_events.h, 1281

CFE_EVS_WriteAppDataFileCmd, 678
 CommandHeader, 678
 Payload, 678

CFE_EVS_WriteAppDataFileCmd_t
 cfe_evs_msg.h, 1313

CFE_EVS_WriteLogFileCmd, 679
 CommandHeader, 679
 Payload, 679

CFE_EVS_WriteLogFileCmd_t
 cfe_evs_msg.h, 1313

CFE_EXECUTIVE_SERVICE
 cfe_error.h, 1105

CFE_FILE_SERVICE
 cfe_error.h, 1105

CFE_FS_BAD_ARGUMENT
 cFE Return Code Defines, 240

CFE_FS_BackgroundFileDumpIsPending

cFE File Utility APIs, 331
CFE_FS_BackgroundFileDumpRequest
 cFE File Utility APIs, 332
CFE_FS_ExtractFilenameFromPath
 cFE File Utility APIs, 332
CFE_FS_FILE_CONTENT_ID
 cfe_fs_extern_typedefs.h, 1145
CFE_FS_FNAME_TOO_LONG
 cFE Return Code Defines, 240
CFE_FS_FileCategory_t
 cfe_fs_api_typedefs.h, 1143
CFE_FS_FileWriteEvent_t
 cfe_fs_api_typedefs.h, 1144
CFE_FS_FileWriteGetData_t
 cfe_fs_api_typedefs.h, 1142
CFE_FS_FileWriteMetaData, 680
 Description, 680
 FileName, 680
 FileSubType, 681
 GetData, 681
 IsPending, 681
 OnEvent, 681
CFE_FS_FileWriteMetaData_t
 cfe_fs_api_typedefs.h, 1142
CFE_FS_FileWriteOnEvent_t
 cfe_fs_api_typedefs.h, 1142
CFE_FS_GetDefaultExtension
 cFE File Utility APIs, 333
CFE_FS_GetDefaultMountPoint
 cFE File Utility APIs, 334
CFE_FS_HDR_DESC_MAX_LEN
 cfe_fs_extern_typedefs.h, 1145
CFE_FS_Header, 682
 ApplicationID, 682
 ContentType, 682
 Description, 683
 Length, 683
 ProcessorID, 683
 SpacecraftID, 683
 SubType, 683
 TimeSeconds, 684
 TimeSubSeconds, 684
CFE_FS_Header_t
 cfe_fs_extern_typedefs.h, 1145
CFE_FS_INVALID_PATH
 cFE Return Code Defines, 240
CFE_FS_InitHeader
 cFE File Header Management APIs, 326
CFE_FS_NOT_IMPLEMENTED
 cFE Return Code Defines, 241
CFE_FS_ParseInputFileName
 cFE File Utility APIs, 334
CFE_FS_ParseInputFileNameEx
 cFE File Utility APIs, 335
CFE_FS_ReadHeader
 cFE File Header Management APIs, 326
CFE_FS_SetTimestamp
 cFE File Header Management APIs, 327
CFE_FS_SubType
 cfe_fs_extern_typedefs.h, 1146
CFE_FS_SubType_Enum_t
 cfe_fs_extern_typedefs.h, 1145
CFE_FS_WriteHeader
 cFE File Header Management APIs, 328
CFE_GENERIC_SERVICE
 cfe_error.h, 1105
CFE_MAJOR_VERSION
 cfe_version.h, 1196
CFE_MAKE_BIG16
 cfe_endian.h, 1098
CFE_MAKE_BIG32
 cfe_endian.h, 1098
CFE_MINOR_VERSION
 cfe_version.h, 1196
CFE_MISSION_ES_APP_TLM_MSG
 sample_mission_cfg.h, 1068
CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN
 sample_mission_cfg.h, 1068
CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
 sample_mission_cfg.h, 1069
CFE_MISSION_ES_CMD_MSG
 sample_mission_cfg.h, 1069
CFE_MISSION_ES_CRC_16
 sample_mission_cfg.h, 1070
CFE_MISSION_ES_CRC_32
 sample_mission_cfg.h, 1070
CFE_MISSION_ES_CRC_8
 sample_mission_cfg.h, 1070
CFE_MISSION_ES_DEFAULT_CRC
 sample_mission_cfg.h, 1070
CFE_MISSION_ES_HK_TLM_MSG
 sample_mission_cfg.h, 1070
CFE_MISSION_ES_MAIN_PERF_ID
 sample_perfids.h, 1088
CFE_MISSION_ES_MAX_APPLICATIONS
 sample_mission_cfg.h, 1071
CFE_MISSION_ES_MEMSTATS_TLM_MSG
 sample_mission_cfg.h, 1071
CFE_MISSION_ES_PERF_EXIT_BIT
 sample_perfids.h, 1088
CFE_MISSION_ES_PERF_MAX_IDS
 sample_mission_cfg.h, 1071
CFE_MISSION_ES_POOL_MAX_BUCKETS
 sample_mission_cfg.h, 1072
CFE_MISSION_ES_SEND_HK_MSG
 sample_mission_cfg.h, 1072
CFE_MISSION_ES_VFS_CMD_MSG
 sample_mission_cfg.h, 1073

CFE_MISSION_EVS_HK_TLM_MSG
sample_mission_cfg.h, 1073

CFE_MISSION_EVS_LONG_EVENT_MSG_MSG
sample_mission_cfg.h, 1073

CFE_MISSION_EVS_MAIN_PERF_ID
sample_perfids.h, 1088

CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
sample_mission_cfg.h, 1073

CFE_MISSION_EVS_SEND_HK_MSG
sample_mission_cfg.h, 1074

CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG
sample_mission_cfg.h, 1074

CFE_MISSION_MAX_API_LEN
sample_mission_cfg.h, 1074

CFE_MISSION_MAX_FILE_LEN
sample_mission_cfg.h, 1074

CFE_MISSION_MAX_PATH_LEN
sample_mission_cfg.h, 1075

CFE_MISSION_REV
cfe_version.h, 1196

CFE_MISSION_SB_ALLSUBS_TLM_MSG
sample_mission_cfg.h, 1075

CFE_MISSION_SB_CMD_MSG
sample_mission_cfg.h, 1076

CFE_MISSION_SB_HK_TLM_MSG
sample_mission_cfg.h, 1076

CFE_MISSION_SB_MAIN_PERF_ID
sample_perfids.h, 1088

CFE_MISSION_SB_MAX_PIPES
sample_mission_cfg.h, 1076

CFE_MISSION_SB_MAX_SB_MSG_SIZE
sample_mission_cfg.h, 1076

CFE_MISSION_SB_MSG_LIM_PERF_ID
sample_perfids.h, 1088

CFE_MISSION_SB_ONESUB_TLM_MSG
sample_mission_cfg.h, 1077

CFE_MISSION_SB_PIPE_OFLOW_PERF_ID
sample_perfids.h, 1089

CFE_MISSION_SB_SEND_HK_MSG
sample_mission_cfg.h, 1077

CFE_MISSION_SB_STATS_TLM_MSG
sample_mission_cfg.h, 1077

CFE_MISSION_SB_SUB_RPT_CTRL_MSG
sample_mission_cfg.h, 1077

CFE_MISSION_TBL_CMD_MSG
sample_mission_cfg.h, 1078

CFE_MISSION_TBL_HK_TLM_MSG
sample_mission_cfg.h, 1078

CFE_MISSION_TBL_MAIN_PERF_ID
sample_perfids.h, 1089

CFE_MISSION_TBL_MAX_FULL_NAME_LEN
sample_mission_cfg.h, 1078

CFE_MISSION_TBL_MAX_NAME_LENGTH
sample_mission_cfg.h, 1078

CFE_MISSION_TBL_REG_TLM_MSG
sample_mission_cfg.h, 1079

CFE_MISSION_TBL_SEND_HK_MSG
sample_mission_cfg.h, 1079

CFE_MISSION_TEST_CMD_MSG
sample_mission_cfg.h, 1079

CFE_MISSION_TEST_HK_TLM_MSG
sample_mission_cfg.h, 1079

CFE_MISSION_TIME_1HZ_CMD_MSG
sample_mission_cfg.h, 1080

CFE_MISSION_TIME_AT_TONE_WAS
sample_mission_cfg.h, 1080

CFE_MISSION_TIME_AT_TONE_WILL_BE
sample_mission_cfg.h, 1080

CFE_MISSION_TIME_CFG_DEFAULT_TAI
sample_mission_cfg.h, 1080

CFE_MISSION_TIME_CFG_DEFAULT_UTC
sample_mission_cfg.h, 1081

CFE_MISSION_TIME_CFG_FAKE_TONE
sample_mission_cfg.h, 1081

CFE_MISSION_TIME_CMD_MSG
sample_mission_cfg.h, 1081

CFE_MISSION_TIME_DATA_CMD_MSG
sample_mission_cfg.h, 1082

CFE_MISSION_TIME_DEF_DELAY_SECS
sample_mission_cfg.h, 1082

CFE_MISSION_TIME_DEF_DELAY_SUBS
sample_mission_cfg.h, 1082

CFE_MISSION_TIME_DEF_LEAPS
sample_mission_cfg.h, 1082

CFE_MISSION_TIME_DEF_MET_SECS
sample_mission_cfg.h, 1082

CFE_MISSION_TIME_DEF_MET_SUBS
sample_mission_cfg.h, 1083

CFE_MISSION_TIME_DEF_STCF_SECS
sample_mission_cfg.h, 1083

CFE_MISSION_TIME_DEF_STCF_SUBS
sample_mission_cfg.h, 1083

CFE_MISSION_TIME_DIAG_TLM_MSG
sample_mission_cfg.h, 1083

CFE_MISSION_TIME_EPOCH_DAY
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_EPOCH_HOUR
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_EPOCH_MICROS
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_EPOCH_MINUTE
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_EPOCH_SECOND
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_EPOCH_YEAR
sample_mission_cfg.h, 1084

CFE_MISSION_TIME_FS_FACTOR
sample_mission_cfg.h, 1085

CFE_MISSION_TIME_HK_TLM_MSG
 sample_mission_cfg.h, 1085
CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID
 sample_perfids.h, 1089
CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID
 sample_perfids.h, 1089
CFE_MISSION_TIME_MAIN_PERF_ID
 sample_perfids.h, 1089
CFE_MISSION_TIME_MAX_ELAPSED
 sample_mission_cfg.h, 1086
CFE_MISSION_TIME_MIN_ELAPSED
 sample_mission_cfg.h, 1086
CFE_MISSION_TIME_SEND_CMD_MSG
 sample_mission_cfg.h, 1086
CFE_MISSION_TIME_SEND_HK_MSG
 sample_mission_cfg.h, 1086
CFE_MISSION_TIME_SEDMET_PERF_ID
 sample_perfids.h, 1090
CFE_MISSION_TIME_TONE1HZISR_PERF_ID
 sample_perfids.h, 1090
CFE_MISSION_TIME_TONE1HZTASK_PERF_ID
 sample_perfids.h, 1090
CFE_MISSION_TIME_TONE_CMD_MSG
 sample_mission_cfg.h, 1087
CFE_MSG_Apld_t
 cfe_msg_api_typedefs.h, 1151
CFE_MSG_BAD_ARGUMENT
 cfe_msg_api_typedefs.h, 1151
CFE_MSG_Checksum_t
 cfe_msg_api_typedefs.h, 1152
CFE_MSG_CommandHeader_t
 cfe_msg_api_typedefs.h, 1152
CFE_MSG_EDSVersion_t
 cfe_msg_api_typedefs.h, 1152
CFE_MSG_Endian
 cfe_msg_api_typedefs.h, 1154
CFE_MSG_Endian_t
 cfe_msg_api_typedefs.h, 1152
CFE_MSG_FcnCode_t
 cfe_msg_api_typedefs.h, 1152
CFE_MSG_GenerateChecksum
 cFE Message Secondary Header APIs, 356
CFE_MSG_GetApld
 cFE Message Primary Header APIs, 338
CFE_MSG_GetEDSVersion
 cFE Message Extended Header APIs, 349
CFE_MSG_GetEndian
 cFE Message Extended Header APIs, 350
CFE_MSG_SetFcnCode
 cFE Message Secondary Header APIs, 357
CFE_MSG_GetHasSecondaryHeader
 cFE Message Primary Header APIs, 339
CFE_MSG_SetHeaderVersion
 cFE Message Primary Header APIs, 340

CFE_MSG_GetMsgId
 cFE Message Id APIs, 362
CFE_MSG_SetMsgTime
 cFE Message Secondary Header APIs, 357
CFE_MSG_SetNextSequenceCount
 cFE Message Primary Header APIs, 340
CFE_MSG_SetPlaybackFlag
 cFE Message Extended Header APIs, 350
CFE_MSG_SetSegmentationFlag
 cFE Message Primary Header APIs, 341
CFE_MSG_SetSequenceCount
 cFE Message Primary Header APIs, 341
CFE_MSG_SetSize
 cFE Message Primary Header APIs, 342
CFE_MSG_SetSubsystem
 cFE Message Extended Header APIs, 351
CFE_MSG_SetSystem
 cFE Message Extended Header APIs, 352
CFE_MSG_SetType
 cFE Message Primary Header APIs, 343
CFE_MSG_SetTypeFromMsgId
 cFE Message Id APIs, 363
CFE_MSG_HeaderVersion_t
 cfe_msg_api_typedefs.h, 1153
CFE_MSG_Init
 cFE Generic Message APIs, 337
CFE_MSG_Message_t
 cfe_msg_api_typedefs.h, 1153
CFE_MSG_NOT_IMPLEMENTED
 cfe_msg_api_typedefs.h, 1151
CFE_MSG_PlaybackFlag
 cfe_msg_api_typedefs.h, 1155
CFE_MSG_PlaybackFlag_t
 cfe_msg_api_typedefs.h, 1153
CFE_MSG_SegmentationFlag
 cfe_msg_api_typedefs.h, 1155
CFE_MSG_SegmentationFlag_t
 cfe_msg_api_typedefs.h, 1153
CFE_MSG_SequenceCount_t
 cfe_msg_api_typedefs.h, 1153
CFE_MSG_SetApld
 cFE Message Primary Header APIs, 343
CFE_MSG_SetEDSVersion
 cFE Message Extended Header APIs, 352
CFE_MSG_SetEndian
 cFE Message Extended Header APIs, 353
CFE_MSG_SetFcnCode
 cFE Message Secondary Header APIs, 358
CFE_MSG_SetHasSecondaryHeader
 cFE Message Primary Header APIs, 345
CFE_MSG_SetHeaderVersion
 cFE Message Primary Header APIs, 345
CFE_MSG_SetMsgId
 cFE Message Id APIs, 363

CFE_MSG_SetMsgTime
 cFE Message Secondary Header APIs, 360

CFE_MSG_SetPlaybackFlag
 cFE Message Extended Header APIs, 354

CFE_MSG_SetSegmentationFlag
 cFE Message Primary Header APIs, 346

CFE_MSG_SetSequenceCount
 cFE Message Primary Header APIs, 347

CFE_MSG_SetSize
 cFE Message Primary Header APIs, 347

CFE_MSG_SetSubsystem
 cFE Message Extended Header APIs, 354

CFE_MSG_SetSystem
 cFE Message Extended Header APIs, 355

CFE_MSG_SetType
 cFE Message Primary Header APIs, 348

CFE_MSG_Size_t
 cfe_msg_api_typedefs.h, 1154

CFE_MSG_Subsystem_t
 cfe_msg_api_typedefs.h, 1154

CFE_MSG_System_t
 cfe_msg_api_typedefs.h, 1154

CFE_MSG_TelemetryHeader_t
 cfe_msg_api_typedefs.h, 1154

CFE_MSG_Type
 cfe_msg_api_typedefs.h, 1155

CFE_MSG_Type_t
 cfe_msg_api_typedefs.h, 1154

CFE_MSG_ValidateChecksum
 cFE Message Secondary Header APIs, 361

CFE_MSG_WRONG_MSG_TYPE
 cfe_msg_api_typedefs.h, 1151

CFE_PLATFORM_CMD_MID_BASE_GLOB
 cpu1_msgids.h, 1004

CFE_PLATFORM_CMD_MID_BASE
 cpu1_msgids.h, 1004

CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
 cpu1_platform_cfg.h, 1012

CFE_PLATFORM_ENDIAN
 cpu1_platform_cfg.h, 1013

CFE_PLATFORM_ES_APP_KILL_TIMEOUT
 cpu1_platform_cfg.h, 1013

CFE_PLATFORM_ES_APP_SCAN_RATE
 cpu1_platform_cfg.h, 1014

CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE
 cpu1_platform_cfg.h, 1014

CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
 cpu1_platform_cfg.h, 1015

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
 cpu1_platform_cfg.h, 1015

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02
 cpu1_platform_cfg.h, 1015

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08
 cpu1_platform_cfg.h, 1016

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14
 cpu1_platform_cfg.h, 1017

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15
 cpu1_platform_cfg.h, 1018

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16
 cpu1_platform_cfg.h, 1018

CFE_PLATFORM_ES_CDS_SIZE
 cpu1_platform_cfg.h, 1018

CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
 cpu1_platform_cfg.h, 1018

CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_←
 FILE
 cpu1_platform_cfg.h, 1019

CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
 cpu1_platform_cfg.h, 1019

CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILE←
 NAME
 cpu1_platform_cfg.h, 1019

CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MO←
 DE
 cpu1_platform_cfg.h, 1020

CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE
 cpu1_platform_cfg.h, 1020

CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
 cpu1_platform_cfg.h, 1021

CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE
 cpu1_platform_cfg.h, 1021

CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE
 cpu1_platform_cfg.h, 1022

CFE_PLATFORM_ES_ER_LOG_ENTRIES
 cpu1_platform_cfg.h, 1022

CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE
 cpu1_platform_cfg.h, 1023

CFE_PLATFORM_ES_MAX_APPLICATIONS

CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
 cpu1_platform_cfg.h, 1023
 CFE_PLATFORM_ES_MAX_BLOCK_SIZE
 cpu1_platform_cfg.h, 1023
 CFE_PLATFORM_ES_MAX_GEN_COUNTERS
 cpu1_platform_cfg.h, 1024
 CFE_PLATFORM_ES_MAX_LIBRARIES
 cpu1_platform_cfg.h, 1024
 CFE_PLATFORM_ES_MAX_MEMORY_POOLS
 cpu1_platform_cfg.h, 1024
 CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS
 cpu1_platform_cfg.h, 1025
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
 cpu1_platform_cfg.h, 1025
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02
 cpu1_platform_cfg.h, 1026
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03
 cpu1_platform_cfg.h, 1026
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04
 cpu1_platform_cfg.h, 1026
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05
 cpu1_platform_cfg.h, 1026
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11
 cpu1_platform_cfg.h, 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN
 cpu1_platform_cfg.h, 1028
 CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRI-
 NG
 cpu1_platform_cfg.h, 1029
 CFE_PLATFORM_ES_NONVOL_STARTUP_FILE
 cpu1_platform_cfg.h, 1029
 CFE_PLATFORM_ES_OBJECT_TABLE_SIZE
 cpu1_platform_cfg.h, 1029
 CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY
 cpu1_platform_cfg.h, 1030
 CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
 cpu1_platform_cfg.h, 1030
 CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE
 cpu1_platform_cfg.h, 1030
 CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
 cpu1_platform_cfg.h, 1031
 CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
 cpu1_platform_cfg.h, 1031
 CFE_PLATFORM_ES_PERF_FILTMASK_ALL
 cpu1_platform_cfg.h, 1031
 CFE_PLATFORM_ES_PERF_FILTMASK_INIT
 cpu1_platform_cfg.h, 1032
 CFE_PLATFORM_ES_PERF_FILTMASK_NONE
 cpu1_platform_cfg.h, 1032
 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
 cpu1_platform_cfg.h, 1032
 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
 cpu1_platform_cfg.h, 1033
 CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
 cpu1_platform_cfg.h, 1033
 CFE_PLATFORM_ES_POOL_MAX_BUCKETS
 cpu1_platform_cfg.h, 1033
 CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
 cpu1_platform_cfg.h, 1034
 CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
 cpu1_platform_cfg.h, 1034
 CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESE-
 RVED
 cpu1_platform_cfg.h, 1034
 CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
 cpu1_platform_cfg.h, 1035
 CFE_PLATFORM_ES_RESET_AREA_SIZE
 cpu1_platform_cfg.h, 1035
 CFE_PLATFORM_ES_START_TASK_PRIORITY
 cpu1_platform_cfg.h, 1036
 CFE_PLATFORM_ES_START_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1036
 CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT-
 _MSEC
 cpu1_platform_cfg.h, 1037
 CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
 cpu1_platform_cfg.h, 1037
 CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
 cpu1_platform_cfg.h, 1038
 CFE_PLATFORM_ES_USER_RESERVED_SIZE
 cpu1_platform_cfg.h, 1038
 CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
 cpu1_platform_cfg.h, 1039
 CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
 cpu1_platform_cfg.h, 1039
 CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
 cpu1_platform_cfg.h, 1040
 CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
 cpu1_platform_cfg.h, 1040

CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT ←
 MODE
 cpu1_platform_cfg.h, 1040
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
 cpu1_platform_cfg.h, 1041
CFE_PLATFORM_EVS_LOG_MAX
 cpu1_platform_cfg.h, 1041
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
 cpu1_platform_cfg.h, 1041
CFE_PLATFORM_EVS_PORT_DEFAULT
 cpu1_platform_cfg.h, 1042
CFE_PLATFORM_EVS_START_TASK_PRIORITY
 cpu1_platform_cfg.h, 1042
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1042
CFE_PLATFORM_SB_BUFS_MEMORY_BYTES
 cpu1_platform_cfg.h, 1043
CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
 cpu1_platform_cfg.h, 1043
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
 cpu1_platform_cfg.h, 1044
CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
 cpu1_platform_cfg.h, 1044
CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME ←
 ME
 cpu1_platform_cfg.h, 1045
CFE_PLATFORM_SB_FILTER_MASK1
 cpu1_platform_cfg.h, 1045
CFE_PLATFORM_SB_FILTER_MASK2
 cpu1_platform_cfg.h, 1045
CFE_PLATFORM_SB_FILTER_MASK3
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTER_MASK4
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTER_MASK5
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTER_MASK6
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTER_MASK7
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTER_MASK8
 cpu1_platform_cfg.h, 1046
CFE_PLATFORM_SB_FILTERED_EVENT1
 cpu1_platform_cfg.h, 1047
CFE_PLATFORM_SB_FILTERED_EVENT2
 cpu1_platform_cfg.h, 1047
CFE_PLATFORM_SB_FILTERED_EVENT3
 cpu1_platform_cfg.h, 1047
CFE_PLATFORM_SB_FILTERED_EVENT4
 cpu1_platform_cfg.h, 1047
CFE_PLATFORM_SB_FILTERED_EVENT5
 cpu1_platform_cfg.h, 1047
CFE_PLATFORM_SB_FILTERED_EVENT6
 cpu1_platform_cfg.h, 1048
CFE_PLATFORM_SB_FILTERED_EVENT7
 cpu1_platform_cfg.h, 1048
CFE_PLATFORM_SB_FILTERED_EVENT8
 cpu1_platform_cfg.h, 1048
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
 cpu1_platform_cfg.h, 1048
CFE_PLATFORM_SB_MAX_BLOCK_SIZE
 cpu1_platform_cfg.h, 1048
CFE_PLATFORM_SB_MAX_DEST_PER_PKT
 cpu1_platform_cfg.h, 1049
CFE_PLATFORM_SB_MAX_MSG_IDS
 cpu1_platform_cfg.h, 1049
CFE_PLATFORM_SB_MAX_PIPES
 cpu1_platform_cfg.h, 1049
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
 cpu1_platform_cfg.h, 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02
 cpu1_platform_cfg.h, 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03
 cpu1_platform_cfg.h, 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09
 cpu1_platform_cfg.h, 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15
 cpu1_platform_cfg.h, 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16
 cpu1_platform_cfg.h, 1053
CFE_PLATFORM_SB_START_TASK_PRIORITY
 cpu1_platform_cfg.h, 1053
CFE_PLATFORM_SB_START_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1053
CFE_PLATFORM_TBL_BUFS_MEMORY_BYTES
 cpu1_platform_cfg.h, 1053
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE
 cpu1_platform_cfg.h, 1054

CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
 cpu1_platform_cfg.h, 1054

CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
 cpu1_platform_cfg.h, 1054

CFE_PLATFORM_TBL_MAX_NUM_HANDLES
 cpu1_platform_cfg.h, 1055

CFE_PLATFORM_TBL_MAX_NUM_TABLES
 cpu1_platform_cfg.h, 1055

CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
 cpu1_platform_cfg.h, 1055

CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
 cpu1_platform_cfg.h, 1056

CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
 cpu1_platform_cfg.h, 1056

CFE_PLATFORM_TBL_START_TASK_PRIORITY
 cpu1_platform_cfg.h, 1057

CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1057

CFE_PLATFORM_TBL_U32FROM4CHARS
 cpu1_platform_cfg.h, 1057

CFE_PLATFORM_TBL_VALID_PRID_1
 cpu1_platform_cfg.h, 1058

CFE_PLATFORM_TBL_VALID_PRID_2
 cpu1_platform_cfg.h, 1058

CFE_PLATFORM_TBL_VALID_PRID_3
 cpu1_platform_cfg.h, 1058

CFE_PLATFORM_TBL_VALID_PRID_4
 cpu1_platform_cfg.h, 1058

CFE_PLATFORM_TBL_VALID_PRID_COUNT
 cpu1_platform_cfg.h, 1059

CFE_PLATFORM_TBL_VALID_SCID_1
 cpu1_platform_cfg.h, 1059

CFE_PLATFORM_TBL_VALID_SCID_2
 cpu1_platform_cfg.h, 1059

CFE_PLATFORM_TBL_VALID_SCID_COUNT
 cpu1_platform_cfg.h, 1060

CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY
 cpu1_platform_cfg.h, 1060

CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1060

CFE_PLATFORM_TIME_CFG_CLIENT
 cpu1_platform_cfg.h, 1060

CFE_PLATFORM_TIME_CFG_LATCH_FLY
 cpu1_platform_cfg.h, 1060

CFE_PLATFORM_TIME_CFG_SERVER
 cpu1_platform_cfg.h, 1061

CFE_PLATFORM_TIME_CFG_SIGNAL
 cpu1_platform_cfg.h, 1061

CFE_PLATFORM_TIME_CFG_SOURCE
 cpu1_platform_cfg.h, 1061

CFE_PLATFORM_TIME_CFG_SRC_GPS
 cpu1_platform_cfg.h, 1062

CFE_PLATFORM_TIME_CFG_SRC_MET
 cpu1_platform_cfg.h, 1062

CFE_PLATFORM_TIME_CFG_SRC_TIME
 cpu1_platform_cfg.h, 1062

CFE_PLATFORM_TIME_CFG_START_FLY
 cpu1_platform_cfg.h, 1063

CFE_PLATFORM_TIME_CFG_TONE_LIMIT
 cpu1_platform_cfg.h, 1063

CFE_PLATFORM_TIME_CFG_VIRTUAL
 cpu1_platform_cfg.h, 1063

CFE_PLATFORM_TIME_MAX_DELTA_SECS
 cpu1_platform_cfg.h, 1064

CFE_PLATFORM_TIME_MAX_DELTA_SUBS
 cpu1_platform_cfg.h, 1064

CFE_PLATFORM_TIME_MAX_LOCAL_SECS
 cpu1_platform_cfg.h, 1065

CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
 cpu1_platform_cfg.h, 1065

CFE_PLATFORM_TIME_START_TASK_PRIORITY
 cpu1_platform_cfg.h, 1065

CFE_PLATFORM_TIME_START_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1065

CFE_PLATFORM_TIME_TONE_TASK_PRIORITY
 cpu1_platform_cfg.h, 1066

CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE
 cpu1_platform_cfg.h, 1066

CFE_PLATFORM_TLM_MID_BASE
 cpu1_msgids.h, 1004

CFE_PSP_AttachExceptions
 cfe_psp.h, 1505

CFE_PSP_Decompress
 cfe_psp.h, 1506

CFE_PSP_ERROR_ADDRESS_MISALIGNED
 cfe_psp.h, 1497

CFE_PSP_ERROR_NOT_IMPLEMENTED
 cfe_psp.h, 1497

CFE_PSP_ERROR_TIMEOUT
 cfe_psp.h, 1497

CFE_PSP_ERROR
 cfe_psp.h, 1497

CFE_PSP_EepromPowerDown
 cfe_psp.h, 1506

CFE_PSP_EepromPowerUp
 cfe_psp.h, 1506

CFE_PSP_EepromWrite16
 cfe_psp.h, 1506

CFE_PSP_EepromWrite32
 cfe_psp.h, 1506

CFE_PSP_EepromWrite8
 cfe_psp.h, 1506

CFE_PSP_EepromWriteDisable
 cfe_psp.h, 1506

CFE_PSP_EepromWriteEnable
 cfe_psp.h, 1507

CFE_PSP_Exception_CopyContext
 cfe_psp.h, 1507

CFE_PSP_Exception_GetCount
 cfe_psp.h, 1507
CFE_PSP_Exception_GetSummary
 cfe_psp.h, 1507
CFE_PSP_FlushCaches
 cfe_psp.h, 1507
CFE_PSP_Get_Dec
 cfe_psp.h, 1507
CFE_PSP_Get_Timebase
 cfe_psp.h, 1508
CFE_PSP_Get_Timer_Tick
 cfe_psp.h, 1508
CFE_PSP_GetBuildNumber
 cfe_psp.h, 1508
CFE_PSP_GetCDSSize
 cfe_psp.h, 1509
CFE_PSP_GetCFETextSegmentInfo
 cfe_psp.h, 1509
CFE_PSP_GetKernelTextSegmentInfo
 cfe_psp.h, 1509
CFE_PSP_GetProcessorId
 cfe_psp.h, 1509
CFE_PSP_GetProcessorName
 cfe_psp.h, 1509
CFE_PSP_GetResetArea
 cfe_psp.h, 1510
CFE_PSP_GetRestartType
 cfe_psp.h, 1510
CFE_PSP_GetSpacecraftId
 cfe_psp.h, 1510
CFE_PSP_GetTime
 cfe_psp.h, 1510
CFE_PSP_GetTimerLow32Rollover
 cfe_psp.h, 1511
CFE_PSP_GetTimerTicksPerSecond
 cfe_psp.h, 1511
CFE_PSP GetUserReservedArea
 cfe_psp.h, 1511
CFE_PSP_GetVersionCodeName
 cfe_psp.h, 1511
CFE_PSP_GetVersionNumber
 cfe_psp.h, 1511
CFE_PSP_GetVersionString
 cfe_psp.h, 1512
CFE_PSP_GetVolatileDiskMem
 cfe_psp.h, 1512
CFE_PSP_INVALID_INT_NUM
 cfe_psp.h, 1497
CFE_PSP_INVALID_MEM_ADDR
 cfe_psp.h, 1497
CFE_PSP_INVALID_MEM_ATTR
 cfe_psp.h, 1498
CFE_PSP_INVALID_MEM_RANGE
 cfe_psp.h, 1498
CFE_PSP_INVALID_MEM_SIZE
 cfe_psp.h, 1498
CFE_PSP_INVALID_MEM_TYPE
 cfe_psp.h, 1498
CFE_PSP_INVALID_MEM_WORDSIZE
 cfe_psp.h, 1498
CFE_PSP_INVALID_MODULE_ID
 cfe_psp.h, 1498
CFE_PSP_INVALID_MODULE_NAME
 cfe_psp.h, 1499
CFE_PSP_INVALID_POINTER
 cfe_psp.h, 1499
CFE_PSP_InitSSR
 cfe_psp.h, 1512
CFE_PSP_MEM_ANY
 cfe_psp.h, 1499
CFE_PSP_MEM_ATTR_READWRITE
 cfe_psp.h, 1499
CFE_PSP_MEM_ATTR_READ
 cfe_psp.h, 1499
CFE_PSP_MEM_ATTR_WRITE
 cfe_psp.h, 1499
CFE_PSP_MEM_EEPROM
 cfe_psp.h, 1500
CFE_PSP_MEM_INVALID
 cfe_psp.h, 1500
CFE_PSP_MEM_RAM
 cfe_psp.h, 1500
CFE_PSP_MEM_SIZE_BYTE
 cfe_psp.h, 1500
CFE_PSP_MEM_SIZE_DWORD
 cfe_psp.h, 1500
CFE_PSP_MEM_SIZE_WORD
 cfe_psp.h, 1500
CFE_PSP_Main
 cfe_psp.h, 1512
CFE_PSP_MemCpy
 cfe_psp.h, 1512
CFE_PSP_MemRangeGet
 cfe_psp.h, 1513
CFE_PSP_MemRangeSet
 cfe_psp.h, 1513
CFE_PSP_MemRanges
 cfe_psp.h, 1513
CFE_PSP_MemRead16
 cfe_psp.h, 1513
CFE_PSP_MemRead32
 cfe_psp.h, 1513
CFE_PSP_MemRead8
 cfe_psp.h, 1514
CFE_PSP_MemSet
 cfe_psp.h, 1514
CFE_PSP_MemValidateRange
 cfe_psp.h, 1514

CFE_PSP_MemWrite16
 cfe_psp.h, 1514
CFE_PSP_MemWrite32
 cfe_psp.h, 1514
CFE_PSP_MemWrite8
 cfe_psp.h, 1514
CFE_PSP_NO_EXCEPTION_DATA
 cfe_psp.h, 1501
CFE_PSP_PANIC_CORE_APP
 cfe_psp.h, 1501
CFE_PSP_PANIC_GENERAL_FAILURE
 cfe_psp.h, 1501
CFE_PSP_PANIC_MEMORY_ALLOC
 cfe_psp.h, 1501
CFE_PSP_PANIC_NONVOL_DISK
 cfe_psp.h, 1501
CFE_PSP_PANIC_STARTUP_SEM
 cfe_psp.h, 1502
CFE_PSP_PANIC_STARTUP
 cfe_psp.h, 1501
CFE_PSP_PANIC_VOLATILE_DISK
 cfe_psp.h, 1502
CFE_PSP_Panic
 cfe_psp.h, 1515
CFE_PSP_PortRead16
 cfe_psp.h, 1515
CFE_PSP_PortRead32
 cfe_psp.h, 1515
CFE_PSP_PortRead8
 cfe_psp.h, 1515
CFE_PSP_PortWrite16
 cfe_psp.h, 1515
CFE_PSP_PortWrite32
 cfe_psp.h, 1515
CFE_PSP_PortWrite8
 cfe_psp.h, 1516
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET
 cfe_psp.h, 1502
CFE_PSP_RST_SUBTYPE_EXCEPTION
 cfe_psp.h, 1502
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND
 cfe_psp.h, 1502
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG
 cfe_psp.h, 1502
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET
 cfe_psp.h, 1503
CFE_PSP_RST_SUBTYPE_MAX
 cfe_psp.h, 1503
CFE_PSP_RST_SUBTYPE_POWER_CYCLE
 cfe_psp.h, 1503
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON
 cfe_psp.h, 1503
CFE_PSP_RST_SUBTYPE_RESET_COMMAND
 cfe_psp.h, 1503

CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET
 cfe_psp.h, 1504
CFE_PSP_RST_TYPE_MAX
 cfe_psp.h, 1504
CFE_PSP_RST_TYPE_POWERON
 cfe_psp.h, 1504
CFE_PSP_SetDefaultExceptionEnvironment
 cfe_psp.h, 1516
CFE_PSP_ReadFromCDS
 cfe_psp.h, 1516
CFE_PSP_Restart
 cfe_psp.h, 1516
CFE_PSP_SOFT_TIMEBASE_NAME
 cfe_psp.h, 1504
CFE_PSP_SUCCESS
 cfe_psp.h, 1505
CFE_PSP_WatchdogDisable
 cfe_psp.h, 1516
CFE_PSP_WatchdogEnable
 cfe_psp.h, 1516
CFE_PSP_WatchdogGet
 cfe_psp.h, 1517
CFE_PSP_WatchdogInit
 cfe_psp.h, 1517
CFE_PSP_WatchdogService
 cfe_psp.h, 1517
CFE_PSP_WatchdogSet
 cfe_psp.h, 1517
CFE_PSP_WriteToCDS
 cfe_psp.h, 1517

CFE_RESOURCEID_MAKE_BASE
 cfe_resourceid_basevalue.h, 1316
CFE_RESOURCEID_MAX
 cfe_resourceid_basevalue.h, 1316
CFE_RESOURCEID_RESERVED
 cfe_resourceid_api_typedefs.h, 1165
CFE_RESOURCEID_SHIFT
 cfe_resourceid_basevalue.h, 1316
CFE_RESOURCEID_TEST_DEFINED
 cfe_resourceid.h, 1158
CFE_RESOURCEID_TEST_EQUAL
 cfe_resourceid.h, 1158
CFE_RESOURCEID_TO ULONG
 cfe_resourceid.h, 1158
CFE_RESOURCEID_UNDEFINED
 cfe_resourceid_api_typedefs.h, 1165
CFE_REVISION
 cfe_version.h, 1196
CFE_Resourceld_Equal
 cfe_resourceid.h, 1159
CFE_Resourceld_FindNext
 cfe_resourceid.h, 1159

CFE_Resourceld_FromInteger
 cfe_resourceid.h, 1160

CFE_Resourceld_GetBase
 cfe_resourceid.h, 1161

CFE_Resourceld_GetSerial
 cfe_resourceid.h, 1161

CFE_Resourceld_IsDefined
 cfe_resourceid.h, 1162

CFE_Resourceld_ToIndex
 cfe_resourceid.h, 1163

CFE_Resourceld_ToInteger
 cfe_resourceid.h, 1164

CFE_SB_ALLSUBS_TLM_MID
 cpu1_msgids.h, 1005

CFE_SB_AllSubscriptionsTlm, 684
 Payload, 685
 TelemetryHeader, 685

CFE_SB_AllSubscriptionsTlm_Payload, 685
 Entries, 686
 Entry, 686
 PktSegment, 686
 TotalSegments, 686

CFE_SB_AllSubscriptionsTlm_Payload_t
 cfe_sb_msg.h, 1354

CFE_SB_AllSubscriptionsTlm_t
 cfe_sb_msg.h, 1354

CFE_SB_AllocateMessageBuffer
 cFE Zero Copy APIs, 381

CFE_SB_BAD_ARGUMENT
 cFE Return Code Defines, 241

CFE_SB_BAD_CMD_CODE_EID
 cfe_sb_events.h, 1319

CFE_SB_BAD_MSGID_EID
 cfe_sb_events.h, 1319

CFE_SB_BAD_PIPEID_EID
 cfe_sb_events.h, 1320

CFE_SB_BUF_ALOC_ERR
 cFE Return Code Defines, 241

CFE_SB_BUFFER_INVALID
 cFE Return Code Defines, 241

CFE_SB_Buffer_t
 cfe_sb_api_typedefs.h, 1173

CFE_SB_CMD0_RCVD_EID
 cfe_sb_events.h, 1320

CFE_SB_CMD1_RCVD_EID
 cfe_sb_events.h, 1320

CFE_SB_CMD_MID
 cpu1_msgids.h, 1005

CFE_SB_CR_PIPE_BAD_ARG_EID
 cfe_sb_events.h, 1321

CFE_SB_CR_PIPE_ERR_EID
 cfe_sb_events.h, 1321

CFE_SB_CR_PIPE_NAME_TAKEN_EID
 cfe_sb_events.h, 1321

CFE_SB_CR_PIPE_NO_FREE_EID
 cfe_sb_events.h, 1322

CFE_SB_CreatePipe
 cFE Pipe Management APIs, 365

CFE_SB_DEFAULT_QOS
 cfe_sb_api_typedefs.h, 1170

CFE_SB_DEL_PIPE_ERR1_EID
 cfe_sb_events.h, 1322

CFE_SB_DEL_PIPE_ERR2_EID
 cfe_sb_events.h, 1322

CFE_SB_DEST_BLK_ERR_EID
 cfe_sb_events.h, 1323

CFE_SB_DISABLE_ROUTE_CC
 cfe_sb_msg.h, 1343

CFE_SB_DISABLE_SUB_REPORTING_CC
 cfe_sb_msg.h, 1343

CFE_SB_DSBL_RTE1_EID
 cfe_sb_events.h, 1323

CFE_SB_DSBL_RTE2_EID
 cfe_sb_events.h, 1323

CFE_SB_DSBL_RTE3_EID
 cfe_sb_events.h, 1324

CFE_SB_DUP_SUBSCRIPTION_EID
 cfe_sb_events.h, 1324

CFE_SB_DeletePipe
 cFE Pipe Management APIs, 366

CFE_SB_DisableRouteCmd_t
 cfe_sb_msg.h, 1354

CFE_SB_DisableSubReportingCmd_t
 cfe_sb_msg.h, 1354

CFE_SB_ENABLE_ROUTE_CC
 cfe_sb_msg.h, 1344

CFE_SB_ENABLE_SUB_REPORTING_CC
 cfe_sb_msg.h, 1345

CFE_SB_ENBL_RTE1_EID
 cfe_sb_events.h, 1324

CFE_SB_ENBL_RTE2_EID
 cfe_sb_events.h, 1325

CFE_SB_ENBL_RTE3_EID
 cfe_sb_events.h, 1325

CFE_SB_EnableRouteCmd_t
 cfe_sb_msg.h, 1354

CFE_SB_EnableSubReportingCmd_t
 cfe_sb_msg.h, 1354

CFE_SB_FILEWRITE_ERR_EID
 cfe_sb_events.h, 1325

CFE_SB_FULL_SUB_PKT_EID
 cfe_sb_events.h, 1326

CFE_SB_GET_BUF_ERR_EID
 cfe_sb_events.h, 1326

CFE_SB_GETPIPEIDBYNAME_EID
 cfe_sb_events.h, 1326

CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID
 cfe_sb_events.h, 1327

CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID
 cfe_sb_events.h, 1327
CFE_SB_GETPIPENAME_EID
 cfe_sb_events.h, 1327
CFE_SB_GETPIPENAME_ID_ERR_EID
 cfe_sb_events.h, 1328
CFE_SB_GETPIPENAME_NULL_PTR_EID
 cfe_sb_events.h, 1328
CFE_SB_GETPIPEOPTS_EID
 cfe_sb_events.h, 1328
CFE_SB_GETPIPEOPTS_ID_ERR_EID
 cfe_sb_events.h, 1329
CFE_SB_GETPIPEOPTS_PTR_ERR_EID
 cfe_sb_events.h, 1329
CFE_SB_GetPipeIdByName
 cFE Pipe Management APIs, 367
CFE_SB_GetPipeName
 cFE Pipe Management APIs, 368
CFE_SB_GetPipeOpts
 cFE Pipe Management APIs, 368
CFE_SB_GetUserData
 cFE Message Characteristics APIs, 384
CFE_SB_GetUserDataLength
 cFE Message Characteristics APIs, 385
CFE_SB_HASHCOLLISION_EID
 cfe_sb_events.h, 1329
CFE_SB_HK_TLM_MID
 cpu1_msgids.h, 1005
CFE_SB_HousekeepingTlm, 687
 Payload, 687
 TelemetryHeader, 687
CFE_SB_HousekeepingTlm_Payload, 688
 CommandCounter, 689
 CommandErrorCounter, 689
 CreatePipeErrorCounter, 689
 DuplicateSubscriptionsCounter, 689
 GetPipeIdByNameErrorCounter, 690
 InternalErrorCounter, 690
 MemInUse, 690
 MemPoolHandle, 690
 MsgLimitErrorCounter, 691
 MsgReceiveErrorCounter, 691
 MsgSendErrorCounter, 691
 NoSubscribersCounter, 691
 PipeOptsErrorCounter, 692
 PipeOverflowErrorCounter, 692
 Spare2Align, 692
 SubscribeErrorCounter, 692
 UnmarkedMem, 693
CFE_SB_HousekeepingTlm_Payload_t
 cfe_sb_msg.h, 1355
CFE_SB_HousekeepingTlm_t
 cfe_sb_msg.h, 1355
CFE_SB_INIT_EID
 cfe_sb_events.h, 1330
CFE_SB_INTERNAL_ERR
 cFE Return Code Defines, 242
CFE_SB_INVALID_MSG_ID
 cfe_sb_api_typedefs.h, 1170
CFE_SB_INVALID_PIPE
 cfe_sb_api_typedefs.h, 1170
CFE_SB_IsValidMsgId
 cFE Message ID APIs, 389
CFE_SB_LEN_ERR_EID
 cfe_sb_events.h, 1330
CFE_SB_MAX_DESTS_MET_EID
 cfe_sb_events.h, 1330
CFE_SB_MAX_DESTS_MET
 cFE Return Code Defines, 242
CFE_SB_MAX_MSGS_MET_EID
 cfe_sb_events.h, 1331
CFE_SB_MAX_MSGS_MET
 cFE Return Code Defines, 242
CFE_SB_MAX_PIPES_MET_EID
 cfe_sb_events.h, 1331
CFE_SB_MAX_PIPES_MET
 cFE Return Code Defines, 242
CFE_SB_MSG_TOO_BIG_EID
 cfe_sb_events.h, 1331
CFE_SB_MSG_TOO_BIG
 cFE Return Code Defines, 243
CFE_SB_MSGID_LIM_ERR_EID
 cfe_sb_events.h, 1332
CFE_SB_MSGID_RESERVED
 cfe_sb_api_typedefs.h, 1171
CFE_SB_MSGID_UNWRAP_VALUE
 cfe_sb_api_typedefs.h, 1171
CFE_SB_MSGID_WRAP_VALUE
 cfe_sb_api_typedefs.h, 1171
CFE_SB_MSGID_C
 cfe_sb_api_typedefs.h, 1170
CFE_SB_MessageStringGet
 cFE Message Characteristics APIs, 385
CFE_SB_MessageStringSet
 cFE Message Characteristics APIs, 386
CFE_SB_Msg, 693
 LongDouble, 694
 LongInt, 694
 Msg, 694
CFE_SB_MsgId_Atom_t
 cfe_sb_extern_typedefs.h, 1175
CFE_SB_MsgId_Equal
 cFE Message ID APIs, 389
CFE_SB_MsgId_t, 694
 Value, 695
CFE_SB_MsgIdToValue
 cFE Message ID APIs, 390
CFE_SB_MsgMapFileEntry, 695

Index, 696
MsgId, 696
CFE_SB_MsgMapFileEntry_t
 cfe_sb_msg.h, 1355
CFE_SB_NO_MESSAGE
 cFE Return Code Defines, 243
CFE_SB_NOOP_CC
 cfe_sb_msg.h, 1346
CFE_SB_NOT_IMPLEMENTED
 cFE Return Code Defines, 243
CFE_SB_NoopCmd_t
 cfe_sb_msg.h, 1355
CFE_SB_ONESUB_TLM_MID
 cpu1_msgids.h, 1005
CFE_SB_PART_SUB_PKT_EID
 cfe_sb_events.h, 1332
CFE_SB_PEND_FOREVER
 cfe_sb_api_typedefs.h, 1172
CFE_SB_PIPE_ADDED_EID
 cfe_sb_events.h, 1332
CFE_SB_PIPE_CR_ERR
 cFE Return Code Defines, 243
CFE_SB_PIPE_DELETED_EID
 cfe_sb_events.h, 1333
CFE_SB_PIPE_RD_ERR
 cFE Return Code Defines, 244
CFE_SB_PIPEID_C
 cfe_sb_api_typedefs.h, 1172
CFE_SB_PIPEOPTS_IGNOREMINE
 cFE SB Pipe options, 392
CFE_SB_POLL
 cfe_sb_api_typedefs.h, 1172
CFE_SB_PipeDepthStats, 696
 CurrentQueueDepth, 697
 MaxQueueDepth, 697
 PeakQueueDepth, 697
 Pipeld, 698
 Spare, 698
CFE_SB_PipeDepthStats_t
 cfe_sb_msg.h, 1355
CFE_SB_Pipeld_ToIndex
 cFE Pipe Management APIs, 370
CFE_SB_Pipeld_t
 cfe_sb_extern_typedefs.h, 1175
CFE_SB_PipeInfoEntry, 698
 AppId, 699
 AppName, 699
 CurrentQueueDepth, 699
 MaxQueueDepth, 700
 Opts, 700
 PeakQueueDepth, 700
 Pipeld, 700
 PipeName, 700
 SendErrors, 701
 Spare, 701
CFE_SB_PipeInfoEntry_t
 cfe_sb_msg.h, 1355
CFE_SB_Q_FULL_ERR_EID
 cfe_sb_events.h, 1333
CFE_SB_Q_RD_ERR_EID
 cfe_sb_events.h, 1333
CFE_SB_Q_WR_ERR_EID
 cfe_sb_events.h, 1334
CFE_SB_Qos_t, 701
 Priority, 702
 Reliability, 702
CFE_SB_QosPriority
 cfe_sb_extern_typedefs.h, 1176
CFE_SB_QosPriority_Enum_t
 cfe_sb_extern_typedefs.h, 1175
CFE_SB_QosReliability
 cfe_sb_extern_typedefs.h, 1176
CFE_SB_QosReliability_Enum_t
 cfe_sb_extern_typedefs.h, 1175
CFE_SB_RCV_BAD_ARG_EID
 cfe_sb_events.h, 1334
CFE_SB_RESET_COUNTERS_CC
 cfe_sb_msg.h, 1347
CFE_SB_ReceiveBuffer
 cFE Send/Receive Message APIs, 378
CFE_SB_ReleaseMessageBuffer
 cFE Zero Copy APIs, 382
CFE_SB_ResetCountersCmd_t
 cfe_sb_msg.h, 1356
CFE_SB_RouteCmd, 702
 CommandHeader, 703
 Payload, 703
CFE_SB_RouteCmd_Payload, 703
 MsgId, 704
 Pipe, 704
 Spare, 704
CFE_SB_RouteCmd_Payload_t
 cfe_sb_msg.h, 1356
CFE_SB_RouteCmd_t
 cfe_sb_msg.h, 1356
CFE_SB_Routeld_Atom_t
 cfe_sb_extern_typedefs.h, 1176
CFE_SB_RoutingFileEntry, 705
 AppName, 705
 MsgCnt, 705
 MsgId, 706
 Pipeld, 706
 PipeName, 706
 State, 706
CFE_SB_RoutingFileEntry_t
 cfe_sb_msg.h, 1356
CFE_SB_SEND_BAD_ARG_EID
 cfe_sb_events.h, 1334

CFE_SB_SEND_HK_MID
 cpu1_msgids.h, 1005
CFE_SB_SEND_INV_MSGID_EID
 cfe_sb_events.h, 1335
CFE_SB_SEND_NO_SUBS_EID
 cfe_sb_events.h, 1335
CFE_SB_SEND_PREV_SUBS_CC
 cfe_sb_msg.h, 1348
CFE_SB_SEND_SB_STATS_CC
 cfe_sb_msg.h, 1349
CFE_SB_SETPPIPEOPTS_EID
 cfe_sb_events.h, 1335
CFE_SB_SETPPIPEOPTS_ID_ERR_EID
 cfe_sb_events.h, 1336
CFE_SB_SETPPIPEOPTS_OWNER_ERR_EID
 cfe_sb_events.h, 1336
CFE_SB_SND_RTG_EID
 cfe_sb_events.h, 1336
CFE_SB_SND_RTG_ERR1_EID
 cfe_sb_events.h, 1337
CFE_SB_SND_STATS_EID
 cfe_sb_events.h, 1337
CFE_SB_STATS_TLM_MID
 cpu1_msgids.h, 1006
CFE_SB_SUB_ARG_ERR_EID
 cfe_sb_events.h, 1337
CFE_SB_SUB_ENTRIES_PER_PKT
 cfe_sb_extern_typedefs.h, 1174
CFE_SB_SUB_INV_CALLER_EID
 cfe_sb_events.h, 1338
CFE_SB_SUB_INV_PIPE_EID
 cfe_sb_events.h, 1338
CFE_SB_SUB_RPT_CTRL_MID
 cpu1_msgids.h, 1006
CFE_SB_SUBSCRIPTION_RCVD_EID
 cfe_sb_events.h, 1338
CFE_SB_SUBSCRIPTION_REMOVED_EID
 cfe_sb_events.h, 1339
CFE_SB_SUBSCRIPTION_RPT_EID
 cfe_sb_events.h, 1339
CFE_SB_SUBSCRIPTION
 cfe_sb_api_typedefs.h, 1173
CFE_SB_SendPrevSubsCmd_t
 cfe_sb_msg.h, 1356
CFE_SB_SendSbStatsCmd_t
 cfe_sb_msg.h, 1357
CFE_SB_SetPipeOpts
 cFE Pipe Management APIs, 371
CFE_SB_SetUserDataLength
 cFE Message Characteristics APIs, 387
CFE_SB_SingleSubscriptionTlm, 707
 Payload, 707
 TelemetryHeader, 707
CFE_SB_SingleSubscriptionTlm_Payload, 707
 MsgId, 708
 Pipe, 708
 Qos, 708
 SubType, 709
CFE_SB_SingleSubscriptionTlm_Payload_t
 cfe_sb_msg.h, 1357
CFE_SB_SingleSubscriptionTlm_t
 cfe_sb_msg.h, 1357
CFE_SB_StatsTlm, 709
 Payload, 709
 TelemetryHeader, 710
CFE_SB_StatsTlm_Payload, 710
 MaxMemAllowed, 711
 MaxMsgIdsAllowed, 711
 MaxPipeDepthAllowed, 711
 MaxPipesAllowed, 712
 MaxSubscriptionsAllowed, 712
 MemInUse, 712
 MsgIdsInUse, 712
 PeakMemInUse, 713
 PeakMsgIdsInUse, 713
 PeakPipesInUse, 713
 PeakSBBuffersInUse, 713
 PeakSubscriptionsInUse, 714
 PipeDepthStats, 714
 PipesInUse, 714
 SBBuffersInUse, 714
 SubscriptionsInUse, 715
CFE_SB_StatsTlm_Payload_t
 cfe_sb_msg.h, 1357
CFE_SB_StatsTlm_t
 cfe_sb_msg.h, 1357
CFE_SB_SubEntries, 715
 MsgId, 716
 Pipe, 716
 Qos, 716
CFE_SB_SubEntries_t
 cfe_sb_msg.h, 1358
CFE_SB_Subscribe
 cFE Message Subscription Control APIs, 372
CFE_SB_SubscribeEx
 cFE Message Subscription Control APIs, 373
CFE_SB_SubscribeLocal
 cFE Message Subscription Control APIs, 374
CFE_SB_TIME_OUT
 cFE Return Code Defines, 244
CFE_SB_TimeStampMsg
 cFE Message Characteristics APIs, 388
CFE_SB_TransmitBuffer
 cFE Zero Copy APIs, 382
CFE_SB_TransmitMsg
 cFE Send/Receive Message APIs, 379
CFE_SB_UNSUB_ARG_ERR_EID
 cfe_sb_events.h, 1339

CFE_SB_UNSUB_INV_CALLER_EID
 cfe_sb_events.h, 1340

CFE_SB_UNSUB_INV_PIPE_EID
 cfe_sb_events.h, 1340

CFE_SB_UNSUB_NO_SUBS_EID
 cfe_sb_events.h, 1340

CFE_SB_UNSUBSCRIPTION
 cfe_sb_api_typedefs.h, 1173

CFE_SB_Unsubscribe
 cFE Message Subscription Control APIs, 375

CFE_SB_UnsubscribeLocal
 cFE Message Subscription Control APIs, 376

CFE_SB_ValueToMsgId
 cFE Message ID APIs, 391

CFE_SB_WRITE_MAP_INFO_CC
 cfe_sb_msg.h, 1350

CFE_SB_WRITE_PIPE_INFO_CC
 cfe_sb_msg.h, 1351

CFE_SB_WRITE_ROUTING_INFO_CC
 cfe_sb_msg.h, 1352

CFE_SB_WRONG_MSG_TYPE
 cFE Return Code Defines, 244

CFE_SB_WriteFileInfoCmd, 717
 CommandHeader, 717
 Payload, 717

CFE_SB_WriteFileInfoCmd_Payload, 718
 Filename, 718

CFE_SB_WriteFileInfoCmd_Payload_t
 cfe_sb_msg.h, 1358

CFE_SB_WriteFileInfoCmd_t
 cfe_sb_msg.h, 1358

CFE_SB_WriteMapInfoCmd_t
 cfe_sb_msg.h, 1358

CFE_SB_WritePipeInfoCmd_t
 cfe_sb_msg.h, 1358

CFE_SB_WriteRoutingInfoCmd_t
 cfe_sb_msg.h, 1358

CFE_SERVICE_BITMASK
 cfe_error.h, 1105

CFE_SEVERITY_BITMASK
 cfe_error.h, 1105

CFE_SEVERITY_ERROR
 cfe_error.h, 1106

CFE_SEVERITY_INFO
 cfe_error.h, 1106

CFE_SEVERITY_SUCCESS
 cfe_error.h, 1106

CFE_SET
 cfe_sb.h, 1168

CFE_SOFTWARE_BUS_SERVICE
 cfe_error.h, 1106

CFE_SRC_VERSION
 cfe_version.h, 1197

CFE_STATUS_BAD_COMMAND_CODE
 cFE Return Code Defines, 244

CFE_STATUS_EXTERNAL_RESOURCE_FAIL
 cFE Return Code Defines, 245

CFE_STATUS_NO_COUNTER_INCREMENT
 cFE Return Code Defines, 245

CFE_STATUS_NOT_IMPLEMENTED
 cFE Return Code Defines, 245

CFE_STATUS_REQUEST_ALREADY_PENDING
 cFE Return Code Defines, 245

CFE_STATUS_UNKNOWN_MSG_ID
 cFE Return Code Defines, 246

CFE_STATUS_WRONG_MSG_LENGTH
 cFE Return Code Defines, 246

CFE_STR_HELPER
 cfe_version.h, 1197

CFE_STR
 cfe_version.h, 1197

CFE_SUCCESS
 cFE Return Code Defines, 246

CFE_Status_t
 cfe_error.h, 1107

CFE_TABLE_SERVICE
 cfe_error.h, 1106

CFE_TBL_ABORT_LOAD_CC
 cfe_tbl_msg.h, 1387

CFE_TBL_ACTIVATE_CC
 cfe_tbl_msg.h, 1388

CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID
 cfe_tbl_events.h, 1361

CFE_TBL_ACTIVATE_ERR_EID
 cfe_tbl_events.h, 1362

CFE_TBL_ASSUMED_VALID_INF_EID
 cfe_tbl_events.h, 1362

CFE_TBL_AbortLoadCmd, 718
 CommandHeader, 719
 Payload, 719

CFE_TBL_AbortLoadCmd_Payload, 719
 TableName, 720

CFE_TBL_AbortLoadCmd_Payload_t
 cfe_tbl_msg.h, 1398

CFE_TBL_AbortLoadCmd_t
 cfe_tbl_msg.h, 1398

CFE_TBL_ActivateCmd, 720
 CommandHeader, 721
 Payload, 721

CFE_TBL_ActivateCmd_Payload, 721
 TableName, 722

CFE_TBL_ActivateCmd_Payload_t
 cfe_tbl_msg.h, 1398

CFE_TBL_ActivateCmd_t
 cfe_tbl_msg.h, 1398

CFE_TBL_BAD_ARGUMENT
 cFE Return Code Defines, 246

CFE_TBL_BAD_TABLE_HANDLE

cfe_tbl_api_typedefs.h, [1180](#)
 CFE_TBL_BufferSelect
 cfe_tbl_extern_typedefs.h, [1182](#)
 CFE_TBL_BufferSelect_Enum_t
 cfe_tbl_extern_typedefs.h, [1182](#)
 CFE_TBL_CC1_ERR_EID
 cfe_tbl_events.h, [1362](#)
 CFE_TBL_CDS_DELETE_ERR_EID
 cfe_tbl_events.h, [1363](#)
 CFE_TBL_CDS_DELETED_INFO_EID
 cfe_tbl_events.h, [1363](#)
 CFE_TBL_CDS_NOT_FOUND_ERR_EID
 cfe_tbl_events.h, [1363](#)
 CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID
 cfe_tbl_events.h, [1364](#)
 CFE_TBL_CMD_MID
 cpu1_msgids.h, [1006](#)
 CFE_TBL_CREATING_DUMP_FILE_ERR_EID
 cfe_tbl_events.h, [1364](#)
 CFE_TBL_CallbackFuncPtr_t
 cfe_tbl_api_typedefs.h, [1180](#)
 CFE_TBL_DELETE_CDS_CC
 cfe_tbl_msg.h, [1389](#)
 CFE_TBL_DUMP_CC
 cfe_tbl_msg.h, [1390](#)
 CFE_TBL_DUMP_PENDING_ERR_EID
 cfe_tbl_events.h, [1364](#)
 CFE_TBL_DUMP_REGISTRY_CC
 cfe_tbl_msg.h, [1391](#)
 CFE_TBL_DelCDSCmd_Payload, [722](#)
 TableName, [723](#)
 CFE_TBL_DelCDSCmd_Payload_t
 cfe_tbl_msg.h, [1398](#)
 CFE_TBL_DeleteCDSCmd, [723](#)
 CommandHeader, [723](#)
 Payload, [724](#)
 CFE_TBL_DeleteCDSCmd_t
 cfe_tbl_msg.h, [1398](#)
 CFE_TBL_DumpCmd, [724](#)
 CommandHeader, [724](#)
 Payload, [725](#)
 CFE_TBL_DumpCmd_Payload, [725](#)
 ActiveTableFlag, [726](#)
 DumpFilename, [726](#)
 TableName, [726](#)
 CFE_TBL_DumpCmd_Payload_t
 cfe_tbl_msg.h, [1399](#)
 CFE_TBL_DumpCmd_t
 cfe_tbl_msg.h, [1399](#)
 CFE_TBL_DumpRegistryCmd, [726](#)
 CommandHeader, [727](#)
 Payload, [727](#)
 CFE_TBL_DumpRegistryCmd_Payload, [727](#)
 DumpFilename, [728](#)

 CFE_TBL_DumpRegistryCmd_Payload_t
 cfe_tbl_msg.h, [1399](#)
 CFE_TBL_DUMP_ONLY
 cFE Return Code Defines, [248](#)
 CFE_TBL_ERR_DUPLICATE_DIFF_SIZE
 cFE Return Code Defines, [248](#)
 CFE_TBL_ERR_DUPLICATE_NOT_OWNED
 cFE Return Code Defines, [248](#)
 CFE_TBL_ERR_FILE_FOR_WRONG_TABLE
 cFE Return Code Defines, [249](#)
 CFE_TBL_ERR_FILE_SIZE_INCONSISTENT
 cFE Return Code Defines, [249](#)
 CFE_TBL_ERR_FILE_TOO_LARGE
 cFE Return Code Defines, [249](#)
 CFE_TBL_ERR_FILENAME_TOO_LONG
 cFE Return Code Defines, [249](#)
 CFE_TBL_ERR_HANDLES_FULL
 cFE Return Code Defines, [250](#)
 CFE_TBL_ERR_ILLEGAL_SRC_TYPE
 cFE Return Code Defines, [250](#)
 CFE_TBL_ERR_INVALID_HANDLE
 cFE Return Code Defines, [250](#)
 CFE_TBL_ERR_INVALID_NAME
 cFE Return Code Defines, [250](#)
 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, [253](#)

CFE_TBL_ERR_NO_TBL_HEADER
 cFE Return Code Defines, 253

CFE_TBL_ERR_PARTIAL_LOAD
 cFE Return Code Defines, 253

CFE_TBL_ERR_REGISTRY_FULL
 cFE Return Code Defines, 253

CFE_TBL_ERR_SHORT_FILE
 cFE Return Code Defines, 254

CFE_TBL_ERR_UNREGISTERED
 cFE Return Code Defines, 254

CFE_TBL_FAIL_HK_SEND_ERR_EID
 cfe_tbl_events.h, 1365

CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID
 cfe_tbl_events.h, 1365

CFE_TBL_FILE_ACCESS_ERR_EID
 cfe_tbl_events.h, 1365

CFE_TBL_FILE_INCOMPLETE_ERR_EID
 cfe_tbl_events.h, 1366

CFE_TBL_FILE_LOADED_INF_EID
 cfe_tbl_events.h, 1366

CFE_TBL_FILE_STD_HDR_ERR_EID
 cfe_tbl_events.h, 1366

CFE_TBL_FILE_SUBTYPE_ERR_EID
 cfe_tbl_events.h, 1367

CFE_TBL_FILE_TBL_HDR_ERR_EID
 cfe_tbl_events.h, 1367

CFE_TBL_FILE_TOO_BIG_ERR_EID
 cfe_tbl_events.h, 1367

CFE_TBL_FILE_TYPE_ERR_EID
 cfe_tbl_events.h, 1368

CFE_TBL_FILEDEF
 cfe_tbl_filedef.h, 1184

CFE_TBL_File_Hdr, 728

- NumBytes, 729
- Offset, 729
- Reserved, 729
- TableName, 729

CFE_TBL_File_Hdr_t
 cfe_tbl_extern_typedefs.h, 1182

CFE_TBL_FileDef, 730

- Description, 730
- ObjectName, 730
- ObjectSize, 730
- TableName, 731
- TgtFilename, 731

CFE_TBL_FileDef_t
 cfe_tbl_filedef.h, 1184

CFE_TBL_GetAddress
 cFE Access Table Content APIs, 406

CFE_TBL_GetAddresses
 cFE Access Table Content APIs, 407

CFE_TBL_GetInfo
 cFE Get Table Information APIs, 411

CFE_TBL_GetStatus

 cFE Get Table Information APIs, 412

CFE_TBL_HANDLE_ACCESS_ERR_EID
 cfe_tbl_events.h, 1368

CFE_TBL_HK_TLM_MID
 cpu1_msgids.h, 1006

CFE_TBL_Handle_t
 cfe_tbl_api_typedefs.h, 1180

CFE_TBL_HousekeepingTlm, 731

- Payload, 732
- TelemetryHeader, 732

CFE_TBL_HousekeepingTlm_Payload, 732

- ActiveBuffer, 733
- ByteAlignPad1, 733
- CommandCounter, 734
- CommandErrorCounter, 734
- FailedValCounter, 734
- LastFileDumped, 734
- LastFileLoaded, 735
- LastTableLoaded, 735
- LastUpdateTime, 735
- LastUpdatedTable, 735
- LastValCrc, 736
- LastValStatus, 736
- LastValTableName, 736
- MemPoolHandle, 736
- NumFreeSharedBufs, 737
- NumLoadPending, 737
- NumTables, 737
- NumValRequests, 737
- SuccessValCounter, 738
- ValidationCounter, 738

CFE_TBL_HousekeepingTlm_Payload_t
 cfe_tbl_msg.h, 1399

CFE_TBL_HousekeepingTlm_t
 cfe_tbl_msg.h, 1399

CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID
 cfe_tbl_events.h, 1368

CFE_TBL_IN_REGISTRY_ERR_EID
 cfe_tbl_events.h, 1369

CFE_TBL_INFO_DUMP_PENDING
 cFE Return Code Defines, 254

CFE_TBL_INFO_NO_UPDATE_PENDING
 cFE Return Code Defines, 254

CFE_TBL_INFO_NO_VALIDATION_PENDING
 cFE Return Code Defines, 255

CFE_TBL_INFO_RECOVERED_TBL
 cFE Return Code Defines, 255

CFE_TBL_INFO_TABLE_LOCKED
 cFE Return Code Defines, 255

CFE_TBL_INFO_UPDATE_PENDING
 cFE Return Code Defines, 255

CFE_TBL_INFO_UPDATED
 cFE Return Code Defines, 256

CFE_TBL_INFO_VALIDATION_PENDING

cFE Return Code Defines, 256
CFE_TBL_INIT_INF_EID
 cfe_tbl_events.h, 1369
CFE_TBL_INTERNAL_ERROR_ERR_EID
 cfe_tbl_events.h, 1369
CFE_TBL_Info, 738
 Crc, 739
 Critical, 739
 DoubleBuffered, 740
 DumpOnly, 740
 FileCreateTimeSecs, 740
 FileCreateTimeSubSecs, 740
 LastFileLoaded, 740
 NumUsers, 741
 Size, 741
 TableLoadedOnce, 741
 TimeOfLastUpdate, 741
 UserDefAddr, 741
CFE_TBL_Info_t
 cfe_tbl_api_typedefs.h, 1180
CFE_TBL_LEN_ERR_EID
 cfe_tbl_events.h, 1370
CFE_TBL_LOAD_ABORT_ERR_EID
 cfe_tbl_events.h, 1370
CFE_TBL_LOAD_ABORT_INF_EID
 cfe_tbl_events.h, 1370
CFE_TBL_LOAD_CC
 cfe_tbl_msg.h, 1392
CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID
 cfe_tbl_events.h, 1371
CFE_TBL_LOAD_FILENAME_LONG_ERR_EID
 cfe_tbl_events.h, 1371
CFE_TBL_LOAD_IN_PROGRESS_ERR_EID
 cfe_tbl_events.h, 1371
CFE_TBL_LOAD_PEND_REQ_INF_EID
 cfe_tbl_events.h, 1372
CFE_TBL_LOAD_SUCCESS_INF_EID
 cfe_tbl_events.h, 1372
CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID
 cfe_tbl_events.h, 1372
CFE_TBL_LOAD_TYPE_ERR_EID
 cfe_tbl_events.h, 1373
CFE_TBL_LOAD_VAL_ERR_EID
 cfe_tbl_events.h, 1373
CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID
 cfe_tbl_events.h, 1373
CFE_TBL_LOADING_PENDING_ERR_EID
 cfe_tbl_events.h, 1374
CFE_TBL_Load
 cFE Manage Table Content APIs, 400
CFE_TBL_LoadCmd, 742
 CommandHeader, 742
 Payload, 742
CFE_TBL_LoadCmd_Payload, 743
LoadFilename, 743
CFE_TBL_LoadCmd_Payload_t
 cfe_tbl_msg.h, 1400
CFE_TBL_LoadCmd_t
 cfe_tbl_msg.h, 1400
CFE_TBL_MAX_FULL_NAME_LEN
 cfe_tbl_api_typedefs.h, 1180
CFE_TBL_MESSAGE_ERROR
 cFE Return Code Defines, 256
CFE_TBL_MID_ERR_EID
 cfe_tbl_events.h, 1374
CFE_TBL_Manage
 cFE Manage Table Content APIs, 401
CFE_TBL_Modified
 cFE Manage Table Content APIs, 402
CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID
 cfe_tbl_events.h, 1374
CFE_TBL_NO_SUCH_TABLE_ERR_EID
 cfe_tbl_events.h, 1375
CFE_TBL_NO_WORK_BUFFERS_ERR_EID
 cfe_tbl_events.h, 1375
CFE_TBL_NOOP_CC
 cfe_tbl_msg.h, 1393
CFE_TBL_NOOP_INF_EID
 cfe_tbl_events.h, 1375
CFE_TBL_NOT_CRITICAL_TBL_ERR_EID
 cfe_tbl_events.h, 1376
CFE_TBL_NOT_IMPLEMENTED
 cFE Return Code Defines, 256
CFE_TBL_NOT_IN_CRIT_REG_ERR_EID
 cfe_tbl_events.h, 1376
CFE_TBL_NoArgsCmd, 744
 CommandHeader, 744
CFE_TBL_NoArgsCmd_t
 cfe_tbl_msg.h, 1400
CFE_TBL_NoopCmd_t
 cfe_tbl_msg.h, 1400
CFE_TBL_NotifyByMessage
 cFE Get Table Information APIs, 413
CFE_TBL_NotifyCmd, 744
 CommandHeader, 745
 Payload, 745
CFE_TBL_NotifyCmd_Payload, 745
 Parameter, 746
CFE_TBL_NotifyCmd_Payload_t
 cfe_tbl_msg.h, 1400
CFE_TBL_NotifyCmd_t
 cfe_tbl_msg.h, 1401
CFE_TBL_OPT_BUFFER_MSK
 cFE Table Type Defines, 415
CFE_TBL_OPT_CRITICAL_MSK
 cFE Table Type Defines, 416
CFE_TBL_OPT_CRITICAL
 cFE Table Type Defines, 415

CFE_TBL_OPT_DBL_BUFFER
 cFE Table Type Defines, [416](#)

CFE_TBL_OPT_DEFAULT
 cFE Table Type Defines, [416](#)

CFE_TBL_OPT_DUMP_ONLY
 cFE Table Type Defines, [416](#)

CFE_TBL_OPT_LD_DMP_MSK
 cFE Table Type Defines, [416](#)

CFE_TBL_OPT_LOAD_DUMP
 cFE Table Type Defines, [417](#)

CFE_TBL_OPT_NOT_CRITICAL
 cFE Table Type Defines, [417](#)

CFE_TBL_OPT_NOT_USR_DEF
 cFE Table Type Defines, [417](#)

CFE_TBL_OPT_SNGL_BUFFER
 cFE Table Type Defines, [417](#)

CFE_TBL_OPT_USR_DEF_ADDR
 cFE Table Type Defines, [417](#)

CFE_TBL_OPT_USR_DEF_MSK
 cFE Table Type Defines, [417](#)

CFE_TBL_OVERWRITE_DUMP_INF_EID
 cfe_tbl_events.h, [1376](#)

CFE_TBL_OVERWRITE_REG_DUMP_INF_EID
 cfe_tbl_events.h, [1377](#)

CFE_TBL_PARTIAL_LOAD_ERR_EID
 cfe_tbl_events.h, [1377](#)

CFE_TBL_PROCESSOR_ID_ERR_EID
 cfe_tbl_events.h, [1377](#)

CFE_TBL_REG_TLM_MID
 cpu1_msgids.h, [1006](#)

CFE_TBL_REGISTER_ERR_EID
 cfe_tbl_events.h, [1378](#)

CFE_TBL_RESET_COUNTERS_CC
 cfe_tbl_msg.h, [1394](#)

CFE_TBL_RESET_INF_EID
 cfe_tbl_events.h, [1378](#)

CFE_TBL_Register
 cFE Registration APIs, [393](#)

CFE_TBL_ReleaseAddress
 cFE Access Table Content APIs, [408](#)

CFE_TBL_ReleaseAddresses
 cFE Access Table Content APIs, [409](#)

CFE_TBL_ResetCountersCmd_t
 cfe_tbl_msg.h, [1401](#)

CFE_TBL_SEND_HK_MID
 cpu1_msgids.h, [1006](#)

CFE_TBL_SEND_REGISTRY_CC
 cfe_tbl_msg.h, [1395](#)

CFE_TBL_SHARE_ERR_EID
 cfe_tbl_events.h, [1378](#)

CFE_TBL_SPACECRAFT_ID_ERR_EID
 cfe_tbl_events.h, [1379](#)

CFE_TBL_SendRegistryCmd, [746](#)
 CommandHeader, [747](#)

Payload, [747](#)

CFE_TBL_SendRegistryCmd_Payload, [747](#)
 TableName, [748](#)

CFE_TBL_SendRegistryCmd_Payload_t
 cfe_tbl_msg.h, [1401](#)

CFE_TBL_SendRegistryCmd_t
 cfe_tbl_msg.h, [1401](#)

CFE_TBL_Share
 cFE Registration APIs, [396](#)

CFE_TBL_SrcEnum
 cfe_tbl_api_typedefs.h, [1181](#)

CFE_TBL_SrcEnum_t
 cfe_tbl_api_typedefs.h, [1181](#)

CFE_TBL_TLM_REG_CMD_INF_EID
 cfe_tbl_events.h, [1379](#)

CFE_TBL_TOO_MANY_DUMPS_ERR_EID
 cfe_tbl_events.h, [1379](#)

CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID
 cfe_tbl_events.h, [1380](#)

CFE_TBL_TableRegistryTlm, [748](#)
 Payload, [749](#)
 TelemetryHeader, [749](#)

CFE_TBL_TableRegistryTlm_t
 cfe_tbl_msg.h, [1401](#)

CFE_TBL_TblRegPacket_Payload, [749](#)
 ActiveBufferAddr, [750](#)
 ByteAlign4, [750](#)
 Crc, [751](#)
 Critical, [751](#)
 DoubleBuffered, [751](#)
 DumpOnly, [751](#)
 FileCreateTimeSecs, [752](#)
 FileCreateTimeSubSecs, [752](#)
 InactiveBufferAddr, [752](#)
 LastFileLoaded, [752](#)
 LoadPending, [753](#)
 Name, [753](#)
 OwnerAppName, [753](#)
 Size, [753](#)
 TableLoadedOnce, [754](#)
 TimeOfLastUpdate, [754](#)
 ValidationFuncPtr, [754](#)

CFE_TBL_TblRegPacket_Payload_t
 cfe_tbl_msg.h, [1401](#)

CFE_TBL_UNREGISTER_ERR_EID
 cfe_tbl_events.h, [1380](#)

CFE_TBL_UNVALIDATED_ERR_EID
 cfe_tbl_events.h, [1380](#)

CFE_TBL_UPDATE_ERR_EID
 cfe_tbl_events.h, [1381](#)

CFE_TBL_UPDATE_SUCCESS_INF_EID
 cfe_tbl_events.h, [1381](#)

CFE_TBL_Unregister
 cFE Registration APIs, [397](#)

CFE_TBL_Update
 cFE Manage Table Content APIs, 403

CFE_TBL_VAL_REQ_MADE_INF_EID
 cfe_tbl_events.h, 1381

CFE_TBL_VALIDATE_CC
 cfe_tbl_msg.h, 1396

CFE_TBL_VALIDATION_ERR_EID
 cfe_tbl_events.h, 1382

CFE_TBL_VALIDATION_INF_EID
 cfe_tbl_events.h, 1382

CFE_TBL_Validate
 cFE Manage Table Content APIs, 404

CFE_TBL_ValidateCmd, 755
 CommandHeader, 755
 Payload, 755

CFE_TBL_ValidateCmd_Payload, 756
 ActiveTableFlag, 756
 TableName, 756

CFE_TBL_ValidateCmd_Payload_t
 cfe_tbl_msg.h, 1402

CFE_TBL_ValidateCmd_t
 cfe_tbl_msg.h, 1402

CFE_TBL_WARN_DUPLICATE
 cFE Return Code Defines, 257

CFE_TBL_WARN_NOT_CRITICAL
 cFE Return Code Defines, 257

CFE_TBL_WARN_PARTIAL_LOAD
 cFE Return Code Defines, 257

CFE_TBL_WARN_SHORT_FILE
 cFE Return Code Defines, 257

CFE_TBL_WRITE_CFE_HDR_ERR_EID
 cfe_tbl_events.h, 1382

CFE_TBL_WRITE_DUMP_INF_EID
 cfe_tbl_events.h, 1383

CFE_TBL_WRITE_REG_DUMP_INF_EID
 cfe_tbl_events.h, 1383

CFE_TBL_WRITE_TBL_HDR_ERR_EID
 cfe_tbl_events.h, 1383

CFE_TBL_WRITE_TBL_IMG_ERR_EID
 cfe_tbl_events.h, 1384

CFE_TBL_WRITE_TBL_REG_ERR_EID
 cfe_tbl_events.h, 1384

CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID
 cfe_tbl_events.h, 1384

CFE_TEST_CMD_MID
 cpu1_msgids.h, 1007

CFE_TEST_HK_TLM_MID
 cpu1_msgids.h, 1007

CFE_TIME_1HZ_CFG_EID
 cfe_time_events.h, 1404

CFE_TIME_1HZ_CMD_MID
 cpu1_msgids.h, 1007

CFE_TIME_1HZ_EID
 cfe_time_events.h, 1404

CFE_TIME_1HzCmd_t
 cfe_time_msg.h, 1436

CFE_TIME_ADD_1HZ_ADJUSTMENT_CC
 cfe_time_msg.h, 1419

CFE_TIME_ADD_ADJUST_CC
 cfe_time_msg.h, 1420

CFE_TIME_ADD_DELAY_CC
 cfe_time_msg.h, 1421

CFE_TIME_Add
 cFE Time Arithmetic APIs, 425

CFE_TIME_Add1HZAdjustmentCmd_t
 cfe_time_msg.h, 1437

CFE_TIME_AddAdjustCmd_t
 cfe_time_msg.h, 1437

CFE_TIME_AddDelayCmd_t
 cfe_time_msg.h, 1437

CFE_TIME_AdjustDirection
 cfe_time_extern_typedefs.h, 1192

CFE_TIME_AdjustDirection_Enum_t
 cfe_time_extern_typedefs.h, 1190

CFE_TIME_BAD_ARGUMENT
 cFE Return Code Defines, 258

CFE_TIME_CALLBACK_NOT_REGISTERED
 cFE Return Code Defines, 258

CFE_TIME_CC_ERR_EID
 cfe_time_events.h, 1404

CFE_TIME_CMD_MID
 cpu1_msgids.h, 1007

CFE_TIME_ClockState
 cfe_time_extern_typedefs.h, 1193

CFE_TIME_ClockState_Enum_t
 cfe_time_extern_typedefs.h, 1190

CFE_TIME_Compare
 cFE Time Arithmetic APIs, 426
 cfe_time_api_typedefs.h, 1188

CFE_TIME_Compare_t
 cfe_time_api_typedefs.h, 1188

CFE_TIME_Copy
 cfe_time.h, 1186

CFE_TIME_DATA_CMD_MID
 cpu1_msgids.h, 1007

CFE_TIME_DELAY_CFG_EID
 cfe_time_events.h, 1405

CFE_TIME_DELAY_EID
 cfe_time_events.h, 1405

CFE_TIME_DELAY_ERR_EID
 cfe_time_events.h, 1405

CFE_TIME_DELTA_CFG_EID
 cfe_time_events.h, 1406

CFE_TIME_DELTA_EID
 cfe_time_events.h, 1406

CFE_TIME_DELTA_ERR_EID
 cfe_time_events.h, 1406

CFE_TIME_DIAG_EID

cfe_time_events.h, 1407
CFE_TIME_DIAG_TLM_MID
cpu1_msgids.h, 1007
CFE_TIME_DiagnosticTlm, 757
 Payload, 757
 TelemetryHeader, 757
CFE_TIME_DiagnosticTlm_Payload, 757
 AtToneDelay, 760
 AtToneLatch, 760
 AtToneLeapSeconds, 760
 AtToneMET, 760
 AtToneSTCF, 760
 ClockFlyState, 761
 ClockSetState, 761
 ClockSignal, 761
 ClockSource, 761
 ClockStateAPI, 762
 ClockStateFlags, 762
 CurrentLatch, 762
 CurrentMET, 762
 CurrentTAI, 763
 CurrentUTC, 763
 DataStoreStatus, 763
 DelayDirection, 763
 Forced2Fly, 764
 LocalIntCounter, 764
 LocalTaskCounter, 764
 MaxElapsed, 764
 MaxLocalClock, 765
 MinElapsed, 765
 OneHzAdjust, 765
 OneHzDirection, 765
 OneTimeAdjust, 766
 OneTimeDirection, 766
 ServerFlyState, 766
 TimeSinceTone, 766
 ToneDataCounter, 767
 ToneDataLatch, 767
 ToneIntCounter, 767
 ToneIntErrorCounter, 767
 ToneMatchCounter, 768
 ToneMatchErrorCounter, 768
 ToneOverLimit, 768
 ToneSignalCounter, 768
 ToneSignalLatch, 769
 ToneTaskCounter, 769
 ToneUnderLimit, 769
 VersionCounter, 769
 VirtualMET, 770
CFE_TIME_DiagnosticTlm_Payload_t
 cfe_time_msg.h, 1437
CFE_TIME_DiagnosticTlm_t
 cfe_time_msg.h, 1437
CFE_TIME_ExternalGPS
 cFE External Time Source APIs, 431
CFE_TIME_ExternalMET
 cFE External Time Source APIs, 432
CFE_TIME_ExternalTime
 cFE External Time Source APIs, 432
CFE_TIME_ExternalTone
 cFE External Time Source APIs, 433
CFE_TIME_FLAG_ADD1HZ
 cFE Clock State Flag Defines, 440
CFE_TIME_FLAG_ADDADJ
 cFE Clock State Flag Defines, 440
CFE_TIME_FLAG_ADDTCL
 cFE Clock State Flag Defines, 441
CFE_TIME_FLAG_CLKSET
 cFE Clock State Flag Defines, 441
CFE_TIME_FLAG_CMDFLY
 cFE Clock State Flag Defines, 441
CFE_TIME_FLAG_FLYING
 cFE Clock State Flag Defines, 441
CFE_TIME_FLAG_GDTONE
 cFE Clock State Flag Defines, 441
CFE_TIME_FLAG_REFERR
 cFE Clock State Flag Defines, 442
CFE_TIME_FLAG_SERVER
 cFE Clock State Flag Defines, 442
CFE_TIME_FLAG_SIGPRI
 cFE Clock State Flag Defines, 442
CFE_TIME_FLAG_SRCINT
 cFE Clock State Flag Defines, 442
CFE_TIME_FLAG_SRVFLY
 cFE Clock State Flag Defines, 442
CFE_TIME_FLAG_UNUSED
 cFE Clock State Flag Defines, 442
CFE_TIME_FLY_OFF_EID
 cfe_time_events.h, 1407
CFE_TIME_FLY_ON_EID
 cfe_time_events.h, 1407
CFE_TIME_FakeToneCmd_t
 cfe_time_msg.h, 1437
CFE_TIME_FlagBit
 cfe_time_extern_typedefs.h, 1193
CFE_TIME_FlagBit_Enum_t
 cfe_time_extern_typedefs.h, 1191
CFE_TIME_FlywheelState
 cfe_time_extern_typedefs.h, 1194
CFE_TIME_FlywheelState_Enum_t
 cfe_time_extern_typedefs.h, 1191
CFE_TIME_GetClockInfo
 cFE Get Time Information APIs, 422
CFE_TIME_GetClockState
 cFE Get Time Information APIs, 422
CFE_TIME_GetLeapSeconds
 cFE Get Time Information APIs, 423
CFE_TIME_GetMETseconds

cFE Get Current Time APIs, [418](#)
CFE_TIME_GetMETsubsecs
 cFE Get Current Time APIs, [419](#)
CFE_TIME_GetMET
 cFE Get Current Time APIs, [418](#)
CFE_TIME_GetSTCF
 cFE Get Time Information APIs, [423](#)
CFE_TIME_GetTAI
 cFE Get Current Time APIs, [419](#)
CFE_TIME_GetTime
 cFE Get Current Time APIs, [420](#)
CFE_TIME_GetUTC
 cFE Get Current Time APIs, [421](#)
CFE_TIME_HK_TLM_MID
 cpu1_msgids.h, [1008](#)
CFE_TIME_HousekeepingTlm, [770](#)
 Payload, [771](#)
 TelemetryHeader, [771](#)
CFE_TIME_HousekeepingTlm_Payload, [771](#)
 ClockStateAPI, [772](#)
 ClockStateFlags, [772](#)
 CommandCounter, [772](#)
 CommandErrorCounter, [773](#)
 LeapSeconds, [773](#)
 Seconds1HzAdj, [773](#)
 SecondsDelay, [773](#)
 SecondsMET, [774](#)
 SecondsSTCF, [774](#)
 Subsecs1HzAdj, [774](#)
 SubsecsDelay, [774](#)
 SubsecsMET, [775](#)
 SubsecsSTCF, [775](#)
CFE_TIME_HousekeepingTlm_Payload_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_HousekeepingTlm_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_ID_ERR_EID
 cfe_time_events.h, [1408](#)
CFE_TIME_INIT_EID
 cfe_time_events.h, [1408](#)
CFE_TIME_INTERNAL_ONLY
 cFE Return Code Defines, [258](#)
CFE_TIME_LEAPS_CFG_EID
 cfe_time_events.h, [1408](#)
CFE_TIME_LEAPS_EID
 cfe_time_events.h, [1409](#)
CFE_TIME_LEN_ERR_EID
 cfe_time_events.h, [1409](#)
CFE_TIME_LeapsCmd_Payload, [775](#)
 LeapSeconds, [776](#)
CFE_TIME_LeapsCmd_Payload_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_Local1HzISR
 cFE Miscellaneous Time APIs, [436](#)
CFE_TIME_MET2SCTime
 cFE Time Conversion APIs, [428](#)
CFE_TIME_MET_CFG_EID
 cfe_time_events.h, [1409](#)
CFE_TIME_MET_EID
 cfe_time_events.h, [1410](#)
CFE_TIME_MET_ERR_EID
 cfe_time_events.h, [1410](#)
CFE_TIME_Micro2SubSecs
 cFE Time Conversion APIs, [428](#)
CFE_TIME_NOOP_CC
 cfe_time_msg.h, [1422](#)
CFE_TIME_NOOP_EID
 cfe_time_events.h, [1410](#)
CFE_TIME_NOT_IMPLEMENTED
 cFE Return Code Defines, [258](#)
CFE_TIME_NoArgsCmd, [776](#)
 CommandHeader, [777](#)
CFE_TIME_NoArgsCmd_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_NoopCmd_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_OUT_OF_RANGE
 cFE Return Code Defines, [259](#)
CFE_TIME_OneHzAdjustmentCmd, [777](#)
 CommandHeader, [777](#)
 Payload, [777](#)
CFE_TIME_OneHzAdjustmentCmd_Payload, [778](#)
 Seconds, [778](#)
 Subseconds, [778](#)
CFE_TIME_OneHzAdjustmentCmd_Payload_t
 cfe_time_msg.h, [1438](#)
CFE_TIME_OneHzAdjustmentCmd_t
 cfe_time_msg.h, [1439](#)
CFE_TIME_PRINTED_STRING_SIZE
 cfe_time_api_typedefs.h, [1187](#)
CFE_TIME_Print
 cFE Miscellaneous Time APIs, [436](#)
CFE_TIME_RESET_COUNTERS_CC
 cfe_time_msg.h, [1423](#)
CFE_TIME_RESET_EID
 cfe_time_events.h, [1411](#)
CFE_TIME_RegisterSynchCallback
 cFE External Time Source APIs, [434](#)
CFE_TIME_ResetCountersCmd_t
 cfe_time_msg.h, [1439](#)
CFE_TIME_SEND_CMD_MID
 cpu1_msgids.h, [1008](#)
CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
 cfe_time_msg.h, [1424](#)
CFE_TIME_SEND_HK_MID
 cpu1_msgids.h, [1008](#)
CFE_TIME_SERVICE
 cfe_error.h, [1107](#)

CFE_TIME_SET_LEAP_SECONDS_CC
 cfe_time_msg.h, 1425

CFE_TIME_SET_MET_CC
 cfe_time_msg.h, 1426

CFE_TIME_SET_SIGNAL_CC
 cfe_time_msg.h, 1427

CFE_TIME_SET_SOURCE_CC
 cfe_time_msg.h, 1428

CFE_TIME_SET_STATE_CC
 cfe_time_msg.h, 1429

CFE_TIME_SET_STCF_CC
 cfe_time_msg.h, 1431

CFE_TIME_SET_TIME_CC
 cfe_time_msg.h, 1432

CFE_TIME_SIGNAL_CFG_EID
 cfe_time_events.h, 1411

CFE_TIME_SIGNAL_EID
 cfe_time_events.h, 1411

CFE_TIME_SIGNAL_ERR_EID
 cfe_time_events.h, 1412

CFE_TIME_SOURCE_CFG_EID
 cfe_time_events.h, 1412

CFE_TIME_SOURCE_EID
 cfe_time_events.h, 1412

CFE_TIME_SOURCE_ERR_EID
 cfe_time_events.h, 1413

CFE_TIME_STATE_EID
 cfe_time_events.h, 1413

CFE_TIME_STATE_ERR_EID
 cfe_time_events.h, 1413

CFE_TIME_STCF_CFG_EID
 cfe_time_events.h, 1414

CFE_TIME_STCF_EID
 cfe_time_events.h, 1414

CFE_TIME_STCF_ERR_EID
 cfe_time_events.h, 1414

CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
 cfe_time_msg.h, 1433

CFE_TIME_SUB_ADJUST_CC
 cfe_time_msg.h, 1434

CFE_TIME_SUB_DELAY_CC
 cfe_time_msg.h, 1435

CFE_TIME_SendDiagnosticCmd_t
 cfe_time_msg.h, 1439

CFE_TIME_SetLeapSecondsCmd, 779
 CommandHeader, 779
 Payload, 779

CFE_TIME_SetLeapSecondsCmd_t
 cfe_time_msg.h, 1439

CFE_TIME_SetMETCmd_t
 cfe_time_msg.h, 1439

CFE_TIME_SetSTCFCmd_t
 cfe_time_msg.h, 1440

CFE_TIME_SetSignalCmd, 780
 CommandHeader, 780
 Payload, 780

CFE_TIME_SetSignalCmd_t
 cfe_time_msg.h, 1439

CFE_TIME_SetSourceCmd, 781
 CommandHeader, 781
 Payload, 781

CFE_TIME_SetSourceCmd_t
 cfe_time_msg.h, 1440

CFE_TIME_SetState
 cfe_time_extern_typedefs.h, 1194

CFE_TIME_SetState_Enum_t
 cfe_time_extern_typedefs.h, 1191

CFE_TIME_SetStateCmd, 782
 CommandHeader, 782
 Payload, 782

CFE_TIME_SetStateCmd_t
 cfe_time_msg.h, 1440

CFE_TIME_SetTimeCmd_t
 cfe_time_msg.h, 1440

CFE_TIME_SignalCmd_Payload, 783
 ToneSource, 783

CFE_TIME_SignalCmd_Payload_t
 cfe_time_msg.h, 1440

CFE_TIME_SourceCmd_Payload, 784
 TimeSource, 784

CFE_TIME_SourceCmd_Payload_t
 cfe_time_msg.h, 1440

CFE_TIME_SourceSelect
 cfe_time_extern_typedefs.h, 1194

CFE_TIME_SourceSelect_Enum_t
 cfe_time_extern_typedefs.h, 1192

CFE_TIME_StateCmd_Payload, 784
 ClockState, 785

CFE_TIME_StateCmd_Payload_t
 cfe_time_msg.h, 1441

CFE_TIME_Sub1HZAdjustmentCmd_t
 cfe_time_msg.h, 1441

CFE_TIME_Sub2MicroSecs
 cFE Time Conversion APIs, 429

CFE_TIME_SubAdjustCmd_t
 cfe_time_msg.h, 1441

CFE_TIME_SubDelayCmd_t
 cfe_time_msg.h, 1441

CFE_TIME_Subtract
 cFE Time Arithmetic APIs, 427

CFE_TIME_SynchCallbackPtr_t
 cfe_time_api_typedefs.h, 1188

CFE_TIME_SysTime, 785
 Seconds, 786
 Subseconds, 786

CFE_TIME_SysTime_t
 cfe_time_extern_typedefs.h, 1192

CFE_TIME_TIME_CFG_EID

cfe_time_events.h, 1415
 CFE_TIME_TIME_EID
 cfe_time_events.h, 1415
 CFE_TIME_TIME_ERR_EID
 cfe_time_events.h, 1415
 CFE_TIME_TONE_CMD_MID
 cpu1_msgids.h, 1008
 CFE_TIME_TOO_MANY_SYNCH_CALLBACKS
 cFE Return Code Defines, 259
 CFE_TIME_TimeCmd, 786
 CommandHeader, 787
 Payload, 787
 CFE_TIME_TimeCmd_Payload, 787
 MicroSeconds, 787
 Seconds, 788
 CFE_TIME_TimeCmd_Payload_t
 cfe_time_msg.h, 1441
 CFE_TIME_TimeCmd_t
 cfe_time_msg.h, 1441
 CFE_TIME_ToneDataCmd, 788
 CommandHeader, 788
 Payload, 789
 CFE_TIME_ToneDataCmd_Payload, 789
 AtToneLeapSeconds, 790
 AtToneMET, 790
 AtToneSTCF, 790
 AtToneState, 790
 CFE_TIME_ToneDataCmd_Payload_t
 cfe_time_msg.h, 1442
 CFE_TIME_ToneDataCmd_t
 cfe_time_msg.h, 1442
 CFE_TIME_ToneSignalCmd_t
 cfe_time_msg.h, 1442
 CFE_TIME_ToneSignalSelect
 cfe_time_extern_typedefs.h, 1194
 CFE_TIME_ToneSignalSelect_Enum_t
 cfe_time_extern_typedefs.h, 1192
 CFE_TIME_UnregisterSynchCallback
 cFE External Time Source APIs, 434
 CFE_TST
 cfe_sb.h, 1168
 CFE_VERSION_STRING
 cfe_version.h, 1197
 CFECoreChecksum
 CFE_ES_HousekeepingTlm_Payload, 607
 CFEMajorVersion
 CFE_ES_HousekeepingTlm_Payload, 607
 CFEMinorVersion
 CFE_ES_HousekeepingTlm_Payload, 607
 CFEMissionRevision
 CFE_ES_HousekeepingTlm_Payload, 607
 CFERevision
 CFE_ES_HousekeepingTlm_Payload, 608
 CFS Health and Safety Command Codes, 206
 HS_DISABLE_ALIVENESS_CC, 206
 HS_DISABLE_APPMON_CC, 207
 HS_DISABLE_CPUHOG_CC, 208
 HS_DISABLE_EVENTMON_CC, 209
 HS_ENABLE_ALIVENESS_CC, 210
 HS_ENABLE_APPMON_CC, 211
 HS_ENABLE_CPUHOG_CC, 212
 HS_ENABLE_EVENTMON_CC, 213
 HS_NOOP_CC, 214
 HS_REPORT_DIAG_CC, 215
 HS_RESET_CC, 216
 HS_RESET_RESETS_PERFORMED_CC, 217
 HS_SET_MAX_RESETS_CC, 218
 HS_SET_UTIL_DIAG_CC, 219
 HS_SET_UTIL_PARAMS_CC, 220
 CFS Health and Safety Command Message IDs, 137
 HS_CMD_MID, 137
 HS_SEND_HK_MID, 137
 HS_WAKEUP_MID, 137
 CFS Health and Safety Command Structures, 204
 CFS Health and Safety Event IDs, 159
 HS_AM_TBL_NULL_ERR_EID, 162
 HS_AMT_LD_ERR_EID, 163
 HS_AMT_REG_ERR_EID, 163
 HS_AMTVAL_ERR_EID, 164
 HS_AMTVAL_INF_EID, 164
 HS_APP_EXIT_EID, 165
 HS_APPMON_APPNAME_DBG_EID, 165
 HS_APPMON_APPNAME_ERR_EID, 166
 HS_APPMON_FAIL_ERR_EID, 166
 HS_APPMON_GETADDR_ERR_EID, 167
 HS_APPMON_MSGACTS_ERR_EID, 167
 HS_APPMON_NOT_RESTARTED_ERR_EID, 168
 HS_APPMON_PROC_ERR_EID, 168
 HS_APPMON_RESTART_ERR_EID, 169
 HS_BADEMT_LONG_UNSUB_EID, 169
 HS_BADEMT_SHORT_UNSUB_EID, 170
 HS_CC_ERR_EID, 170
 HS_CDS_CORRUPT_ERR_EID, 171
 HS_CDS_RESTORE_ERR_EID, 171
 HS_CPUMON_HOGGING_ERR_EID, 172
 HS_CR_CHILD_TASK_ERR_EID, 172
 HS_CR_CMD_PIPE_ERR_EID, 173
 HS_CR_EVENT_PIPE_ERR_EID, 173
 HS_CR_SYNC_CALLBACK_ERR_EID, 174
 HS_CR_WAKEUP_PIPE_ERR_EID, 174
 HS_CUSTOM_INIT_ERR_EID, 175
 HS_DISABLE_ALIVENESS_DBG_EID, 175
 HS_DISABLE_APPMON_DBG_EID, 176
 HS_DISABLE_APPMON_ERR_EID, 176
 HS_DISABLE_CPUHOG_DBG_EID, 177
 HS_DISABLE_EVENTMON_DBG_EID, 177
 HS_DISABLE_EVENTMON_ERR_EID, 178
 HS_EM_TBL_NULL_ERR_EID, 178

HS_EMT_LD_ERR_EID, 179
HS_EMT_REG_ERR_EID, 179
HS_EMTVAL_ERR_EID, 180
HS_EMTVAL_INF_EID, 180
HS_ENABLE_ALIVENESS_DBG_EID, 181
HS_ENABLE_APPMON_DBG_EID, 181
HS_ENABLE_CPUHOG_DBG_EID, 182
HS_ENABLE_EVENTMON_DBG_EID, 182
HS_EVENTMON_DELETE_ERR_EID, 183
HS_EVENTMON_GETADDR_ERR_EID, 183
HS_EVENTMON_LONG_SUB_EID, 184
HS_EVENTMON_LONG_UNSUB_EID, 184
HS_EVENTMON_MSGACTS_ERR_EID, 185
HS_EVENTMON_NOT_DELETED_ERR_EID, 185
HS_EVENTMON_NOT_RESTARTED_ERR_EID, 186
HS_EVENTMON_PROC_ERR_EID, 186
HS_EVENTMON_RESTART_ERR_EID, 187
HS_EVENTMON_SHORT_SUB_EID, 187
HS_EVENTMON_SHORT_UNSUB_EID, 188
HS_EXECOUNT_GETADDR_ERR_EID, 188
HS_HKREQ_LEN_ERR_EID, 189
HS_HKREQ_RESOURCE_DBG_EID, 189
HS_INIT_EID, 190
HS_LEN_ERR_EID, 190
HS_MA_TBL_NULL_ERR_EID, 191
HS_MAT_LD_ERR_EID, 191
HS_MAT_REG_ERR_EID, 192
HS_MATVAL_ERR_EID, 192
HS_MATVAL_INF_EID, 193
HS_MID_ERR_EID, 193
HS_MSGACTS_GETADDR_ERR_EID, 194
HS_NOOP_INF_EID, 194
HS_RESET_DBG_EID, 195
HS_RESET_LIMIT_ERR_EID, 195
HS_RESET_RESETS_DBG_EID, 196
HS_SET_MAX_RESETS_DBG_EID, 196
HS_SET_UTIL_DIAG_DBG_EID, 197
HS_SET_UTIL_PARAMS_DBG_EID, 197
HS_SET_UTIL_PARAMS_ERR_EID, 198
HS_SUB_CMD_ERR_EID, 198
HS_SUB_LONG_EVSEERR_EID, 199
HS_SUB_REQ_ERR_EID, 199
HS_SUB_SHORT_EVSEERR_EID, 200
HS_SUB_WAKEUP_ERR_EID, 200
HS_UTIL_DIAG_REPORT_EID, 201
HS_XC_TBL_NULL_ERR_EID, 201
HS_XCT_LD_ERR_EID, 202
HS_XCT_REG_ERR_EID, 202
HS_XCTVAL_ERR_EID, 203
HS_XCTVAL_INF_EID, 203
CFS Health and Safety Mission Configuration, 135
HS_APPMAIN_PERF_ID, 135
HS_IDLETASK_PERF_ID, 135
CFS Health and Safety Platform Configuration, 139
HS_ALIVENESS_DEFAULT_STATE, 140
HS_AMT_FILENAME, 141
HS_APP_NAME, 141
HS_APPMON_DEFAULT_STATE, 141
HS_CMD_PIPE_DEPTH, 142
HS_CPU_ALIVE_PERIOD, 142
HS_CPU_ALIVE_STRING, 142
HS_CPUHOG_DEFAULT_STATE, 143
HS_EMT_FILENAME, 143
HS_EVENT_PIPE_DEPTH, 143
HS_EVENTMON_DEFAULT_STATE, 144
HS_IDLE_TASK_FLAGS, 144
HS_IDLE_TASK_NAME, 144
HS_IDLE_TASK_PRIORITY, 145
HS_IDLE_TASK_STACK_PTR, 145
HS_IDLE_TASK_STACK_SIZE, 145
HS_MAT_FILENAME, 146
HS_MAX_EXEC_CNT_SLOTS, 146
HS_MAX_MONITORED_APPS, 146
HS_MAX_MONITORED_EVENTS, 147
HS_MAX_MSG_ACT_SIZE, 147
HS_MAX_MSG_ACT_TYPES, 148
HS_MAX_RESTART_ACTIONS, 148
HS_MISSION_REV, 149
HS_POST_PROCESSING_DELAY, 149
HS_RESET_TASK_DELAY, 150
HS_STARTUP_SYNC_TIMEOUT, 150
HS_UTIL_AVERAGE_NUM_INTERVAL, 151
HS_UTIL_CALLS_PER_MARK, 151
HS_UTIL_CONV_DIV, 152
HS_UTIL_CONV_MULT1, 152
HS_UTIL_CONV_MULT2, 153
HS_UTIL_CYCLES_PER_INTERVAL, 153
HS_UTIL_DIAG_MASK, 154
HS_UTIL_HOGGING_TIMEOUT, 154
HS_UTIL_PEAK_NUM_INTERVAL, 154
HS_UTIL_PER_INTERVAL_HOGGING, 155
HS_UTIL_PER_INTERVAL_TOTAL, 155
HS_UTIL_TIME_DIAG_ARRAY_LENGTH, 156
HS_UTIL_TIME_DIAG_ARRAY_MASK, 156
HS_UTIL_TIME_DIAG_ARRAY_POWER, 156
HS_WAKEUP_PIPE_DEPTH, 156
HS_WAKEUP_TIMEOUT, 157
HS_WATCHDOG_TIMEOUT_VALUE, 157
HS_XCT_FILENAME, 158
CFS Health and Safety Telemetry, 205
CFS Health and Safety Telemetry Message IDs, 138
HS_HK_TLM_MID, 138
CFS Health and Safety Version, 222
HS_MAJOR_VERSION, 222
HS_MINOR_VERSION, 222
HS_REVISION, 222
ccsds_hdr.h

CCSDS_ExtendedHeader_t, 1314
 CCSDS_PrimaryHeader_t, 1314
CdsName
 CFE_ES_DeleteCDSCmd_Payload, 600
 cfe/cmake/sample_defs/cpu1_msghdr.h, 1000
 cfe/cmake/sample_defs/cpu1_platform_cfg.h, 1009
 cfe/cmake/sample_defs/sample_mission_cfg.h, 1066
 cfe/cmake/sample_defs/sample_perfids.h, 1087
 cfe/docs/src/cfe_api.dox, 1091
 cfe/docs/src/cfe_es.dox, 1091
 cfe/docs/src/cfe_evs.dox, 1091
 cfe/docs/src/cfe_frontpage.dox, 1091
 cfe/docs/src/cfe_glossary.dox, 1091
 cfe/docs/src/cfe_sb.dox, 1091
 cfe/docs/src/cfe_tbl.dox, 1091
 cfe/docs/src/cfe_time.dox, 1091
 cfe/docs/src/cfe_xref.dox, 1091
 cfe/docs/src/cfs_versions.dox, 1091
 cfe/modules/core_api/fsw/inc/cfe.h, 1091
 cfe/modules/core_api/fsw/inc/cfe_config.h, 1092
 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h, 1096
 cfe/modules/core_api/fsw/inc/cfe_endian.h, 1098
 cfe/modules/core_api/fsw/inc/cfe_error.h, 1098
 cfe/modules/core_api/fsw/inc/cfe_es.h, 1107
 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h, 1111
 cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h, 1118
 cfe/modules/core_api/fsw/inc/cfe_evs.h, 1130
 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h, 1132
 cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h, 1135
 cfe/modules/core_api/fsw/inc/cfe_fs.h, 1140
 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h, 1141
 cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h, 1144
 cfe/modules/core_api/fsw/inc/cfe_msg.h, 1147
 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h, 1149
 cfe/modules/core_api/fsw/inc/cfe_resourceid.h, 1157
 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h, 1164
 cfe/modules/core_api/fsw/inc/cfe_sb.h, 1166
 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h, 1169
 cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h, 1173
 cfe/modules/core_api/fsw/inc/cfe_tbl.h, 1177
 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h, 1178
 cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h, 1181
 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h, 1183
 cfe/modules/core_api/fsw/inc/cfe_time.h, 1185
 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h, 1187
 cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h, 1189
 cfe/modules/core_api/fsw/inc/cfe_version.h, 1195
 cfe/modules/es/fsw/inc/cfe_es_events.h, 1198
 cfe/modules/es/fsw/inc/cfe_es_msg.h, 1231
 cfe/modules/evs/fsw/inc/cfe_evs_events.h, 1267
 cfe/modules/evs/fsw/inc/cfe_evs_msg.h, 1281
 cfe/modules/msg/fsw/inc/ccsds_hdr.h, 1313
 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h, 1314
 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h, 1315
 cfe/modules/sb/fsw/inc/cfe_sb_events.h, 1317
 cfe/modules/sb/fsw/inc/cfe_sb_msg.h, 1341
 cfe/modules/tbl/fsw/inc/cfe_tbl_events.h, 1359
 cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h, 1385
 cfe/modules/time/fsw/inc/cfe_time_events.h, 1402
 cfe/modules/time/fsw/inc/cfe_time_msg.h, 1416
cfe_config.h
 CFE_Config_GetIdByName, 1092
 CFE_Config_GetName, 1093
 CFE_Config_GetObjPointer, 1093
 CFE_Config_GetString, 1094
 CFE_Config_GetValue, 1094
 CFE_Config_IterateAll, 1096
cfe_config_api_typedefs.h
 CFE_CONFIGID_UNDEFINED, 1097
 CFE_CONFIGID_C, 1097
 CFE_Config_Callback_t, 1097
 CFE_ConfigId_t, 1097
cfe_endian.h
 CFE_MAKE_BIG16, 1098
 CFE_MAKE_BIG32, 1098
cfe_error.h
 CFE_EVENTS_SERVICE, 1105
 CFE_EXECUTIVE_SERVICE, 1105
 CFE_FILE_SERVICE, 1105
 CFE_GENERIC_SERVICE, 1105
 CFE_SERVICE_BITMASK, 1105
 CFE_SEVERITY_BITMASK, 1105
 CFE_SEVERITY_ERROR, 1106
 CFE_SEVERITY_INFO, 1106
 CFE_SEVERITY_SUCCESS, 1106
 CFE_SOFTWARE_BUS_SERVICE, 1106
 CFE_Status_t, 1107
 CFE_TABLE_SERVICE, 1106
 CFE_TIME_SERVICE, 1107
cfe_es.h
 CFE_ES_DBIT, 1110
 CFE_ES_DTEST, 1110

CFE_ES_TEST_LONG_MASK, 1111
OS_PRINTF, 1111
cfe_es_api_typedefs.h
CFE_ES_APP_RESTART, 1113
CFE_ES_APPID_UNDEFINED, 1113
CFE_ES_APPID_C, 1113
CFE_ES_CDS_BAD_HANDLE, 1113
CFE_ES_CDSHANDLE_C, 1113
CFE_ES_COUNTERID_UNDEFINED, 1114
CFE_ES_COUNTERID_C, 1114
CFE_ES_ChildTaskMainFuncPtr_t, 1116
CFE_ES_LIBID_UNDEFINED, 1114
CFE_ES_LIBID_C, 1114
CFE_ES_LibraryEntryFuncPtr_t, 1117
CFE_ES_MEMHANDLE_UNDEFINED, 1114
CFE_ES_MEMHANDLE_C, 1114
CFE_ES_MEMPOOLBUF_C, 1115
CFE_ES_MemPoolBuf_t, 1117
CFE_ES_NO_MUTEX, 1115
CFE_ES_PoolAlign_t, 1117
CFE_ES_STATIC_POOL_TYPE, 1115
CFE_ES_StackPointer_t, 1117
CFE_ES_TASK_STACK_ALLOCATE, 1115
CFE_ES_TASKID_UNDEFINED, 1116
CFE_ES_TASKID_C, 1116
CFE_ES_TaskEntryFuncPtr_t, 1118
CFE_ES_USE_MUTEX, 1116
cfe_es_events.h
CFE_ES_ALL_APPS_EID, 1201
CFE_ES_BOOT_ERR_EID, 1201
CFE_ES_BUILD_INF_EID, 1202
CFE_ES_CC1_ERR_EID, 1202
CFE_ES_CDS_DELETE_ERR_EID, 1203
CFE_ES_CDS_DELETE_TBL_ERR_EID, 1203
CFE_ES_CDS_DELETED_INFO_EID, 1203
CFE_ES_CDS_DUMP_ERR_EID, 1204
CFE_ES_CDS_NAME_ERR_EID, 1204
CFE_ES_CDS_OWNER_ACTIVE_EID, 1204
CFE_ES_CDS_REG_DUMP_INF_EID, 1205
CFE_ES_CDS_REGISTER_ERR_EID, 1205
CFE_ES_CREATING_CDS_DUMP_ERR_EID, 1205
CFE_ES_ERLOG1_INF_EID, 1206
CFE_ES_ERLOG2_EID, 1206
CFE_ES_ERLOG2_ERR_EID, 1206
CFE_ES_ERLOG_PENDING_ERR_EID, 1207
CFE_ES_ERR_SYSLOGMODE_EID, 1207
CFE_ES_ERREXIT_APP_ERR_EID, 1207
CFE_ES_ERREXIT_APP_INF_EID, 1208
CFE_ES_EXIT_APP_ERR_EID, 1208
CFE_ES_EXIT_APP_INF_EID, 1209
CFE_ES_FILEWRITE_ERR_EID, 1209
CFE_ES_INIT_INF_EID, 1209
CFE_ES_INITSTATS_INF_EID, 1210
CFE_ES_INVALID_POOL_HANDLE_ERR_EID, 1210
CFE_ES_LEN_ERR_EID, 1210
CFE_ES_MID_ERR_EID, 1211
CFE_ES_NOOP_INF_EID, 1211
CFE_ES_ONE_APP_EID, 1211
CFE_ES_ONE_APPID_ERR_EID, 1212
CFE_ES_ONE_ERR_EID, 1212
CFE_ES_OSCREATE_ERR_EID, 1212
CFE_ES_PCR_ERR1_EID, 1213
CFE_ES_PCR_ERR2_EID, 1213
CFE_ES_PERF_DATAWRITTEN_EID, 1213
CFE_ES_PERF_FILTMSKCMD_EID, 1214
CFE_ES_PERF_FILTMSKERR_EID, 1214
CFE_ES_PERF_LOG_ERR_EID, 1214
CFE_ES_PERF_STARTCMD_EID, 1215
CFE_ES_PERF_STARTCMD_ERR_EID, 1215
CFE_ES_PERF_STARTCMD_TRIG_ERR_EID, 1215
CFE_ES_PERF_STOPCMD_EID, 1216
CFE_ES_PERF_STOPCMD_ERR2_EID, 1216
CFE_ES_PERF_TRIGMSKCMD_EID, 1216
CFE_ES_PERF_TRIGMSKERR_EID, 1217
CFE_ES_RELOAD_APP_DBG_EID, 1217
CFE_ES_RELOAD_APP_ERR1_EID, 1217
CFE_ES_RELOAD_APP_ERR2_EID, 1218
CFE_ES_RELOAD_APP_ERR3_EID, 1218
CFE_ES_RELOAD_APP_ERR4_EID, 1218
CFE_ES_RELOAD_APP_INF_EID, 1219
CFE_ES_RESET_INF_EID, 1219
CFE_ES_RESET_PR_COUNT_EID, 1219
CFE_ES_RESTART_APP_DBG_EID, 1220
CFE_ES_RESTART_APP_ERR1_EID, 1220
CFE_ES_RESTART_APP_ERR2_EID, 1220
CFE_ES_RESTART_APP_ERR3_EID, 1221
CFE_ES_RESTART_APP_ERR4_EID, 1221
CFE_ES_RESTART_APP_INF_EID, 1221
CFE_ES_SET_MAX_PR_COUNT_EID, 1222
CFE_ES_START_ERR_EID, 1222
CFE_ES_START_EXC_ACTION_ERR_EID, 1222
CFE_ES_START_INF_EID, 1223
CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID, 1223
CFE_ES_START_INVALID_FILENAME_ERR_EID, 1223
CFE_ES_START_NULL_APP_NAME_ERR_EID, 1224
CFE_ES_START_PRIORITY_ERR_EID, 1224
CFE_ES_STOP_DBG_EID, 1224
CFE_ES_STOP_ERR1_EID, 1225
CFE_ES_STOP_ERR2_EID, 1225
CFE_ES_STOP_ERR3_EID, 1225
CFE_ES_STOP_INF_EID, 1226
CFE_ES_SYSLOG1_INF_EID, 1226

CFE_ES_SYSLOG2_EID, [1226](#)
 CFE_ES_SYSLOG2_ERR_EID, [1227](#)
 CFE_ES_SYSLOGMODE_EID, [1227](#)
 CFE_ES_TASKINFO_EID, [1227](#)
 CFE_ES_TASKINFO_OSCREATE_ERR_EID, [1228](#)
 CFE_ES_TASKINFO_WR_ERR_EID, [1228](#)
 CFE_ES_TASKINFO_WRHDR_ERR_EID, [1228](#)
 CFE_ES_TASKWR_ERR_EID, [1229](#)
 CFE_ES_TLM_POOL_STATS_INFO_EID, [1229](#)
 CFE_ES_VERSION_INF_EID, [1229](#)
 CFE_ES_WRHDR_ERR_EID, [1230](#)
 CFE_ES_WRITE_CFE_HDR_ERR_EID, [1230](#)
cfe_es_extern_typedefs.h
 CFE_ES_AppId_t, [1121](#)
 CFE_ES_AppInfo_t, [1121](#)
 CFE_ES_AppState, [1127](#)
 CFE_ES_AppState_Enum_t, [1121](#)
 CFE_ES_AppType, [1127](#)
 CFE_ES_AppType_Enum_t, [1121](#)
 CFE_ES_BlockStats_t, [1122](#)
 CFE_ES_CDCHandle_t, [1122](#)
 CFE_ES_CDSRegDumpRec_t, [1122](#)
 CFE_ES_CounterId_t, [1122](#)
 CFE_ES_ExceptionAction, [1128](#)
 CFE_ES_ExceptionAction_Enum_t, [1123](#)
 CFE_ES_LibId_t, [1123](#)
 CFE_ES_LogEntryType, [1128](#)
 CFE_ES_LogEntryType_Enum_t, [1123](#)
 CFE_ES_LogMode, [1128](#)
 CFE_ES_LogMode_Enum_t, [1123](#)
 CFE_ES_MEMADDRESS_C, [1120](#)
 CFE_ES_MEMOFFSET_C, [1120](#)
 CFE_ES_MemAddress_t, [1124](#)
 CFE_ES_MemHandle_t, [1124](#)
 CFE_ES_MemOffset_t, [1124](#)
 CFE_ES_MemPoolStats_t, [1125](#)
 CFE_ES_RunStatus, [1129](#)
 CFE_ES_RunStatus_Enum_t, [1125](#)
 CFE_ES_SystemState, [1129](#)
 CFE_ES_SystemState_Enum_t, [1125](#)
 CFE_ES_TaskId_t, [1126](#)
 CFE_ES_TaskInfo_t, [1126](#)
 CFE_ES_TaskPriority_Atom_t, [1126](#)
cfe_es_msg.h
 CFE_ES_AppNameCmd_Payload_t, [1259](#)
 CFE_ES_AppNameCmd_t, [1259](#)
 CFE_ES_AppReloadCmd_Payload_t, [1259](#)
 CFE_ES_CLEAR_ER_LOG_CC, [1235](#)
 CFE_ES_CLEAR_SYSLOG_CC, [1235](#)
 CFE_ES_ClearERLogCmd_t, [1259](#)
 CFE_ES_ClearSysLogCmd_t, [1259](#)
 CFE_ES_DELETE_CDS_CC, [1236](#)
 CFE_ES_DUMP_CDS_REGISTRY_CC, [1237](#)
 CFE_ES_DeleteCDSCmd_Payload_t, [1259](#)
 CFE_ES_DeleteCDSCmd_t, [1259](#)
 CFE_ES_DumpCDSRegistryCmd_Payload_t, [1260](#)
 CFE_ES_DumpCDSRegistryCmd_t, [1260](#)
 CFE_ES_FileNameCmd_Payload_t, [1260](#)
 CFE_ES_FileNameCmd_t, [1260](#)
 CFE_ES_HousekeepingTlm_Payload_t, [1260](#)
 CFE_ES_HousekeepingTlm_t, [1260](#)
 CFE_ES_MemStatsTlm_t, [1261](#)
 CFE_ES_NOOP_CC, [1238](#)
 CFE_ES_NoArgsCmd_t, [1261](#)
 CFE_ES_NoopCmd_t, [1261](#)
 CFE_ES_OVER_WRITE_SYSLOG_CC, [1239](#)
 CFE_ES_OneAppTlm_Payload_t, [1261](#)
 CFE_ES_OneAppTlm_t, [1261](#)
 CFE_ES_OverWriteSysLogCmd_Payload_t, [1261](#)
 CFE_ES_OverWriteSysLogCmd_t, [1262](#)
 CFE_ES_PoolStatsTlm_Payload_t, [1262](#)
 CFE_ES_QUERY_ALL_CC, [1240](#)
 CFE_ES_QUERY_ALL_TASKS_CC, [1241](#)
 CFE_ES_QUERY_ONE_CC, [1242](#)
 CFE_ES_QueryAllCmd_t, [1262](#)
 CFE_ES_QueryAllTasksCmd_t, [1262](#)
 CFE_ES_QueryOneCmd_t, [1262](#)
 CFE_ES_RELOAD_APP_CC, [1243](#)
 CFE_ES_RESET_COUNTERS_CC, [1244](#)
 CFE_ES_RESET_PR_COUNT_CC, [1245](#)
 CFE_ES_RESTART_APP_CC, [1246](#)
 CFE_ES_RESTART_CC, [1247](#)
 CFE_ES_ReloadAppCmd_t, [1262](#)
 CFE_ES_ResetCountersCmd_t, [1263](#)
 CFE_ES_ResetPRCountCmd_t, [1263](#)
 CFE_ES_RestartAppCmd_t, [1263](#)
 CFE_ES_RestartCmd_Payload_t, [1263](#)
 CFE_ES_RestartCmd_t, [1263](#)
 CFE_ES_SEND_MEM_POOL_STATS_CC, [1248](#)
 CFE_ES_SET_MAX_PR_COUNT_CC, [1249](#)
 CFE_ES_SET_PERF_FILTER_MASK_CC, [1250](#)
 CFE_ES_SET_PERF_TRIGGER_MASK_CC, [1251](#)
 CFE_ES_START_APP_CC, [1252](#)
 CFE_ES_START_PERF_DATA_CC, [1253](#)
 CFE_ES_STOP_APP_CC, [1254](#)
 CFE_ES_STOP_PERF_DATA_CC, [1255](#)
 CFE_ES_SendMemPoolStatsCmd_Payload_t, [1263](#)
 CFE_ES_SendMemPoolStatsCmd_t, [1264](#)
 CFE_ES_SetMaxPRCountCmd_Payload_t, [1264](#)
 CFE_ES_SetMaxPRCountCmd_t, [1264](#)
 CFE_ES_SetPerfFilterMaskCmd_Payload_t, [1264](#)
 CFE_ES_SetPerfFilterMaskCmd_t, [1264](#)
 CFE_ES_SetPerfTrigMaskCmd_Payload_t, [1265](#)
 CFE_ES_SetPerfTriggerMaskCmd_t, [1264](#)
 CFE_ES_StartAppCmd_Payload_t, [1265](#)
 CFE_ES_StartAppCmd_t, [1265](#)
 CFE_ES_StartPerfCmd_Payload_t, [1265](#)
 CFE_ES_StartPerfDataCmd_t, [1265](#)

CFE_ES_StopAppCmd_t, 1265
CFE_ES_StopPerfCmd_Payload_t, 1266
CFE_ES_StopPerfDataCmd_t, 1266
CFE_ES_WRITE_ER_LOG_CC, 1256
CFE_ES_WRITE_SYSLOG_CC, 1257
CFE_ES_WriteERLogCmd_t, 1266
CFE_ES_WriteSysLogCmd_t, 1266

cfe_evs.h
CFE_EVS_Send, 1130
CFE_EVS_SendCrit, 1131
CFE_EVS_SendDbg, 1131
CFE_EVS_SendErr, 1131
CFE_EVS_SendInfo, 1131

cfe_evs_api_typedefs.h
CFE_EVS_BinFilter_t, 1135
CFE_EVS_EVERY_FOURTH_ONE, 1133
CFE_EVS_EVERY_OTHER_ONE, 1133
CFE_EVS_EVERY_OTHER_TWO, 1133
CFE_EVS_FIRST_16_STOP, 1133
CFE_EVS_FIRST_32_STOP, 1133
CFE_EVS_FIRST_4_STOP, 1133
CFE_EVS_FIRST_64_STOP, 1134
CFE_EVS_FIRST_8_STOP, 1134
CFE_EVS_FIRST_ONE_STOP, 1134
CFE_EVS_FIRST_TWO_STOP, 1134
CFE_EVS_NO_FILTER, 1134

cfe_evs_events.h
CFE_EVS_ADDFILTER_EID, 1268
CFE_EVS_DELFILTER_EID, 1268
CFE_EVS_DISAPPENTTYPE_EID, 1269
CFE_EVS_DISAPPEVT_EID, 1269
CFE_EVS_DISEVTTYPE_EID, 1269
CFE_EVS_DISPRT_EID, 1270
CFE_EVS_ENAAPPEVT_EID, 1270
CFE_EVS_ENAAPPEVTTYPE_EID, 1270
CFE_EVS_ENAEVTTYPE_EID, 1271
CFE_EVS_ENAPORT_EID, 1271
CFE_EVS_ERR_APPNOREGS_EID, 1271
CFE_EVS_ERR_CC_EID, 1272
CFE_EVS_ERR_CRDATFILE_EID, 1272
CFE_EVS_ERR_CRLOGFILE_EID, 1272
CFE_EVS_ERR_EVTIDNOREGS_EID, 1273
CFE_EVS_ERR_ILLAPPIDRANGE_EID, 1273
CFE_EVS_ERR_ILLEGALFMTMOD_EID, 1273
CFE_EVS_ERR_INVALID_BITMASK_EID, 1274
CFE_EVS_ERR_LOGMODE_EID, 1274
CFE_EVS_ERR_MAXREGSFILTER_EID, 1274
CFE_EVS_ERR_MSGID_EID, 1275
CFE_EVS_ERR_NOAPPIDFOUND_EID, 1275
CFE_EVS_ERR_UNREGISTERED_EVS_APP, 1275
CFE_EVS_ERR_WRDATFILE_EID, 1276
CFE_EVS_ERR_WRLOGFILE_EID, 1276
CFE_EVS_EVT_FILTERED_EID, 1276

CFE_EVS_FILTER_MAX_EID, 1277
CFE_EVS_LEN_ERR_EID, 1277
CFE_EVS_LOGMODE_EID, 1277
CFE_EVS_NOOP_EID, 1278
CFE_EVS_RSTALLFILTER_EID, 1278
CFE_EVS_RSTCNT_EID, 1278
CFE_EVS_RSTEVCNT_EID, 1279
CFE_EVS_RSTFILTER_EID, 1279
CFE_EVS_SETEVTFMTMOD_EID, 1279
CFE_EVS_SETFILTERMSK_EID, 1280
CFE_EVS_STARTUP_EID, 1280
CFE_EVS_WRDAT_EID, 1280
CFE_EVS_WRLOG_EID, 1281

cfe_evs_extern_typedefs.h
CFE_EVS_EventFilter, 1137
CFE_EVS_EventFilter_Enum_t, 1136
CFE_EVS_EventOutput, 1137
CFE_EVS_EventOutput_Enum_t, 1136
CFE_EVS_EventType, 1139
CFE_EVS_EventType_Enum_t, 1136
CFE_EVS_LogMode, 1139
CFE_EVS_LogMode_Enum_t, 1136
CFE_EVS_MsgFormat, 1139
CFE_EVS_MsgFormat_Enum_t, 1137

cfe_evs_msg.h
CFE_EVS_ADD_EVENT_FILTER_CC, 1285
CFE_EVS_AddEventFilterCmd_t, 1306
CFE_EVS_AppDataCmd_Payload_t, 1306
CFE_EVS_AppNameBitMaskCmd_Payload_t, 1307
CFE_EVS_AppNameBitMaskCmd_t, 1307
CFE_EVS_AppNameCmd_Payload_t, 1307
CFE_EVS_AppNameCmd_t, 1307
CFE_EVS_AppNameEventIDCmd_Payload_t, 1307
CFE_EVS_AppNameEventIDCmd_t, 1307
CFE_EVS_AppNameEventIDMaskCmd_Payload_t, 1308
CFE_EVS_AppNameEventIDMaskCmd_t, 1308
CFE_EVS_AppTlmData_t, 1308
CFE_EVS_BitMaskCmd_Payload_t, 1308
CFE_EVS_BitMaskCmd_t, 1308
CFE_EVS_CLEAR_LOG_CC, 1285
CFE_EVS_CRITICAL_BIT, 1286
CFE_EVS_ClearLogCmd_t, 1308
CFE_EVS_DEBUG_BIT, 1286
CFE_EVS_DELETE_EVENT_FILTER_CC, 1287
CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, 1287
CFE_EVS_DISABLE_APP_EVENTS_CC, 1288
CFE_EVS_DISABLE_EVENT_TYPE_CC, 1289
CFE_EVS_DISABLE_PORTS_CC, 1290
CFE_EVS_DeleteEventFilterCmd_t, 1309
CFE_EVS_DisableAppEventTypeCmd_t, 1309
CFE_EVS_DisableAppEventsCmd_t, 1309
CFE_EVS_DisableEventTypeCmd_t, 1309

CFE_EVS_DisablePortsCmd_t, [1309](#)
 CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, [1291](#)
 CFE_EVS_ENABLE_APP_EVENTS_CC, [1292](#)
 CFE_EVS_ENABLE_EVENT_TYPE_CC, [1293](#)
 CFE_EVS_ENABLE_PORTS_CC, [1294](#)
 CFE_EVS_ERROR_BIT, [1295](#)
 CFE_EVS_EnableAppEventTypeCmd_t, [1310](#)
 CFE_EVS_EnableAppEventsCmd_t, [1309](#)
 CFE_EVS_EnableEventTypeCmd_t, [1310](#)
 CFE_EVS_EnablePortsCmd_t, [1310](#)
 CFE_EVS_HousekeepingTlm_Payload_t, [1310](#)
 CFE_EVS_HousekeepingTlm_t, [1310](#)
 CFE_EVS_INFORMATION_BIT, [1296](#)
 CFE_EVS_LogFileCmd_Payload_t, [1310](#)
 CFE_EVS_LongEventTlm_Payload_t, [1311](#)
 CFE_EVS_LongEventTlm_t, [1311](#)
 CFE_EVS_NOOP_CC, [1296](#)
 CFE_EVS_NoArgsCmd_t, [1311](#)
 CFE_EVS_NoopCmd_t, [1311](#)
 CFE_EVS_PORT1_BIT, [1296](#)
 CFE_EVS_PORT2_BIT, [1297](#)
 CFE_EVS_PORT3_BIT, [1297](#)
 CFE_EVS_PORT4_BIT, [1297](#)
 CFE_EVS_PacketID_t, [1311](#)
 CFE_EVS_RESET_ALL_FILTERS_CC, [1297](#)
 CFE_EVS_RESET_APP_COUNTER_CC, [1298](#)
 CFE_EVS_RESET_COUNTERS_CC, [1299](#)
 CFE_EVS_RESET_FILTER_CC, [1300](#)
 CFE_EVS_ResetAllFiltersCmd_t, [1311](#)
 CFE_EVS_ResetAppCounterCmd_t, [1311](#)
 CFE_EVS_ResetCountersCmd_t, [1312](#)
 CFE_EVS_ResetFilterCmd_t, [1312](#)
 CFE_EVS_SET_EVENT_FORMAT_MODE_CC,
 [1301](#)
 CFE_EVS_SET_FILTER_CC, [1302](#)
 CFE_EVS_SET_LOG_MODE_CC, [1303](#)
 CFE_EVS_SetEventFormatMode_Payload_t, [1312](#)
 CFE_EVS_SetEventFormatModeCmd_t, [1312](#)
 CFE_EVS_SetFilterCmd_t, [1312](#)
 CFE_EVS_SetLogMode_Payload_t, [1312](#)
 CFE_EVS_SetLogModeCmd_t, [1313](#)
 CFE_EVS_ShortEventTlm_Payload_t, [1313](#)
 CFE_EVS_ShortEventTlm_t, [1313](#)
 CFE_EVS_WRITE_APP_DATA_FILE_CC, [1304](#)
 CFE_EVS_WRITE_LOG_DATA_FILE_CC, [1305](#)
 CFE_EVS_WriteAppDataFileCmd_t, [1313](#)
 CFE_EVS_WriteLogDataFileCmd_t, [1313](#)
cfe_fs_api_typedefs.h
 CFE_FS_FileCategory_t, [1143](#)
 CFE_FS_FileWriteEvent_t, [1144](#)
 CFE_FS_FileWriteGetData_t, [1142](#)
 CFE_FS_FileWriteMetaData_t, [1142](#)
 CFE_FS_FileWriteOnEvent_t, [1142](#)
cfe_fs_extern_typedefs.h
 CFE_FS_FILE_CONTENT_ID, [1145](#)
 CFE_FS_HDR_DESC_MAX_LEN, [1145](#)
 CFE_FS_Header_t, [1145](#)
 CFE_FS_SubType, [1146](#)
 CFE_FS_SubType_Enum_t, [1145](#)
cfe_msg_api_typedefs.h
 CFE_MSG_Apld_t, [1151](#)
 CFE_MSG_BAD_ARGUMENT, [1151](#)
 CFE_MSG_Checksum_t, [1152](#)
 CFE_MSG_CommandHeader_t, [1152](#)
 CFE_MSG_EDSVersion_t, [1152](#)
 CFE_MSG_Endian, [1154](#)
 CFE_MSG_Endian_t, [1152](#)
 CFE_MSG_FcnCode_t, [1152](#)
 CFE_MSG_HeaderVersion_t, [1153](#)
 CFE_MSG_Message_t, [1153](#)
 CFE_MSG_NOT_IMPLEMENTED, [1151](#)
 CFE_MSG_PlaybackFlag, [1155](#)
 CFE_MSG_PlaybackFlag_t, [1153](#)
 CFE_MSG_SegmentationFlag, [1155](#)
 CFE_MSG_SegmentationFlag_t, [1153](#)
 CFE_MSG_SequenceCount_t, [1153](#)
 CFE_MSG_Size_t, [1154](#)
 CFE_MSG_Subsystem_t, [1154](#)
 CFE_MSG_System_t, [1154](#)
 CFE_MSG_TelemetryHeader_t, [1154](#)
 CFE_MSG_Type, [1155](#)
 CFE_MSG_Type_t, [1154](#)
 CFE_MSG_WRONG_MSG_TYPE, [1151](#)
cfe_psp.h
 BUFF_SIZE, [1496](#)
 CFE_PSP_AttachExceptions, [1505](#)
 CFE_PSP_Decompress, [1506](#)
 CFE_PSP_ERROR_ADDRESS_MISALIGNED,
 [1497](#)
 CFE_PSP_ERROR_NOT_IMPLEMENTED, [1497](#)
 CFE_PSP_ERROR_TIMEOUT, [1497](#)
 CFE_PSP_ERROR, [1497](#)
 CFE_PSP_EepromPowerDown, [1506](#)
 CFE_PSP_EepromPowerUp, [1506](#)
 CFE_PSP_EepromWrite16, [1506](#)
 CFE_PSP_EepromWrite32, [1506](#)
 CFE_PSP_EepromWrite8, [1506](#)
 CFE_PSP_EepromWriteDisable, [1506](#)
 CFE_PSP_EepromWriteEnable, [1507](#)
 CFE_PSP_Exception_CopyContext, [1507](#)
 CFE_PSP_Exception_GetCount, [1507](#)
 CFE_PSP_Exception_GetSummary, [1507](#)
 CFE_PSP_FlushCaches, [1507](#)
 CFE_PSP_Get_Dec, [1507](#)
 CFE_PSP_Get_Timebase, [1508](#)
 CFE_PSP_Get_Timer_Tick, [1508](#)
 CFE_PSP_GetBuildNumber, [1508](#)
 CFE_PSP_GetCDSSize, [1509](#)

CFE_PSP_GetCFETextSegmentInfo, 1509
CFE_PSP_GetKernelTextSegmentInfo, 1509
CFE_PSP_GetProcessorId, 1509
CFE_PSP_GetProcessorName, 1509
CFE_PSP_GetResetArea, 1510
CFE_PSP_GetRestartType, 1510
CFE_PSP_GetSpacecraftId, 1510
CFE_PSP_GetTime, 1510
CFE_PSP_GetTimerLow32Rollover, 1511
CFE_PSP_GetTimerTicksPerSecond, 1511
CFE_PSP.GetUserReservedArea, 1511
CFE_PSP_GetVersionCodeName, 1511
CFE_PSP_GetVersionNumber, 1511
CFE_PSP_GetVersionString, 1512
CFE_PSP_GetVolatileDiskMem, 1512
CFE_PSP_INVALID_INT_NUM, 1497
CFE_PSP_INVALID_MEM_ADDR, 1497
CFE_PSP_INVALID_MEM_ATTR, 1498
CFE_PSP_INVALID_MEM_RANGE, 1498
CFE_PSP_INVALID_MEM_SIZE, 1498
CFE_PSP_INVALID_MEM_TYPE, 1498
CFE_PSP_INVALID_MEM_WORDSIZE, 1498
CFE_PSP_INVALID_MODULE_ID, 1498
CFE_PSP_INVALID_MODULE_NAME, 1499
CFE_PSP_INVALID_POINTER, 1499
CFE_PSP_InitSSR, 1512
CFE_PSP_MEM_ANY, 1499
CFE_PSP_MEM_ATTR_READWRITE, 1499
CFE_PSP_MEM_ATTR_READ, 1499
CFE_PSP_MEM_ATTR_WRITE, 1499
CFE_PSP_MEM_EEPROM, 1500
CFE_PSP_MEM_INVALID, 1500
CFE_PSP_MEM_RAM, 1500
CFE_PSP_MEM_SIZE_BYTE, 1500
CFE_PSP_MEM_SIZE_DWORD, 1500
CFE_PSP_MEM_SIZE_WORD, 1500
CFE_PSP_Main, 1512
CFE_PSP_MemCpy, 1512
CFE_PSP_MemRangeGet, 1513
CFE_PSP_MemRangeSet, 1513
CFE_PSP_MemRanges, 1513
CFE_PSP_MemRead16, 1513
CFE_PSP_MemRead32, 1513
CFE_PSP_MemRead8, 1514
CFE_PSP_MemSet, 1514
CFE_PSP_MemValidateRange, 1514
CFE_PSP_MemWrite16, 1514
CFE_PSP_MemWrite32, 1514
CFE_PSP_MemWrite8, 1514
CFE_PSP_NO_EXCEPTION_DATA, 1501
CFE_PSP_PANIC_CORE_APP, 1501
CFE_PSP_PANIC_GENERAL_FAILURE, 1501
CFE_PSP_PANIC_MEMORY_ALLOC, 1501
CFE_PSP_PANIC_NONVOL_DISK, 1501
CFE_PSP_PANIC_STARTUP_SEM, 1502
CFE_PSP_PANIC_STARTUP, 1501
CFE_PSP_PANIC_VOLATILE_DISK, 1502
CFE_PSP_Panic, 1515
CFE_PSP_PortRead16, 1515
CFE_PSP_PortRead32, 1515
CFE_PSP_PortRead8, 1515
CFE_PSP_PortWrite16, 1515
CFE_PSP_PortWrite32, 1515
CFE_PSP_PortWrite8, 1516
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET, 1502
CFE_PSP_RST_SUBTYPE_EXCEPTION, 1502
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND, 1502
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG, 1502
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET, 1503
CFE_PSP_RST_SUBTYPE_MAX, 1503
CFE_PSP_RST_SUBTYPE_POWER_CYCLE, 1503
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON, 1503
CFE_PSP_RST_SUBTYPE_RESET_COMMAND, 1503
CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET, 1504
CFE_PSP_RST_TYPE_MAX, 1504
CFE_PSP_RST_TYPE_POWERON, 1504
CFE_PSP_RST_TYPE_PROCESSOR, 1504
CFE_PSP_ReadFromCDS, 1516
CFE_PSP_Restart, 1516
CFE_PSP_SOFT_TIMEBASE_NAME, 1504
CFE_PSP_SUCCESS, 1505
CFE_PSP_SetDefaultExceptionEnvironment, 1516
CFE_PSP_WatchdogDisable, 1516
CFE_PSP_WatchdogEnable, 1516
CFE_PSP_WatchdogGet, 1517
CFE_PSP_WatchdogInit, 1517
CFE_PSP_WatchdogService, 1517
CFE_PSP_WatchdogSet, 1517
CFE_PSP_WriteToCDS, 1517
SIZE_BYTE, 1505
SIZE_HALF, 1505
SIZE_WORD, 1505
cfe_resourceid.h
CFE_RESOURCEID_TEST_DEFINED, 1158
CFE_RESOURCEID_TEST_EQUAL, 1158
CFE_RESOURCEID_TO ULONG, 1158
CFE_Resourceld_Equal, 1159
CFE_Resourceld_FindNext, 1159
CFE_Resourceld_FromInteger, 1160
CFE_Resourceld_GetBase, 1161
CFE_Resourceld_GetSerial, 1161
CFE_Resourceld_IsDefined, 1162

CFE_Resourceld_ToIndex, 1163
 CFE_Resourceld_ToInteger, 1164
 cfe_resourceid_api_typedefs.h
 CFE_RESOURCEID_RESERVED, 1165
 CFE_RESOURCEID_UNDEFINED, 1165
 cfe_resourceid_basevalue.h
 CFE_RESOURCEID_MAKE_BASE, 1316
 CFE_RESOURCEID_MAX, 1316
 CFE_RESOURCEID_SHIFT, 1316
 cfe_sb.h
 CFE_BIT, 1167
 CFE_CLR, 1168
 CFE_SET, 1168
 CFE_TST, 1168
 cfe_sb_api_typedefs.h
 CFE_SB_Buffer_t, 1173
 CFE_SB_DEFAULT_QOS, 1170
 CFE_SB_INVALID_MSG_ID, 1170
 CFE_SB_INVALID_PIPE, 1170
 CFE_SB_MSGID_RESERVED, 1171
 CFE_SB_MSGID_UNWRAP_VALUE, 1171
 CFE_SB_MSGID_WRAP_VALUE, 1171
 CFE_SB_MSGID_C, 1170
 CFE_SB_PEND_FOREVER, 1172
 CFE_SB PIPEID_C, 1172
 CFE_SB_POLL, 1172
 CFE_SB_SUBSCRIPTION, 1173
 CFE_SB_UNSUBSCRIPTION, 1173
 cfe_sb_events.h
 CFE_SB_BAD_CMD_CODE_EID, 1319
 CFE_SB_BAD_MSGID_EID, 1319
 CFE_SB_BAD_PIPEID_EID, 1320
 CFE_SB_CMD0_RCVD_EID, 1320
 CFE_SB_CMD1_RCVD_EID, 1320
 CFE_SB_CR_PIPE_BAD_ARG_EID, 1321
 CFE_SB_CR_PIPE_ERR_EID, 1321
 CFE_SB_CR_PIPE_NAME_TAKEN_EID, 1321
 CFE_SB_CR_PIPE_NO_FREE_EID, 1322
 CFE_SB_DEL_PIPE_ERR1_EID, 1322
 CFE_SB_DEL_PIPE_ERR2_EID, 1322
 CFE_SB_DEST_BLK_ERR_EID, 1323
 CFE_SB_DSBBL RTE1_EID, 1323
 CFE_SB_DSBBL RTE2_EID, 1323
 CFE_SB_DSBBL RTE3_EID, 1324
 CFE_SB_DUP_SUBSCRIPT_EID, 1324
 CFE_SB_ENBL RTE1_EID, 1324
 CFE_SB_ENBL RTE2_EID, 1325
 CFE_SB_ENBL RTE3_EID, 1325
 CFE_SB_FILEWRITE_ERR_EID, 1325
 CFE_SB_FULL_SUB_PKT_EID, 1326
 CFE_SB_GET_BUF_ERR_EID, 1326
 CFE_SB_GETPIPEIDBYNAME_EID, 1326
 CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID, 1327
 CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID, 1327
 CFE_SB_GETPIPEOPTS_EID, 1328
 CFE_SB_GETPIPEOPTS_ID_ERR_EID, 1328
 CFE_SB_GETPIPEOPTS_PTR_ERR_EID, 1328
 CFE_SB_HASHCOLLISION_EID, 1329
 CFE_SB_INIT_EID, 1330
 CFE_SB_LEN_ERR_EID, 1330
 CFE_SB_MAX_DESTS_MET_EID, 1330
 CFE_SB_MAX_MSGS_MET_EID, 1331
 CFE_SB_MAX_PIPES_MET_EID, 1331
 CFE_SB_MSG_TOO_BIG_EID, 1331
 CFE_SB_MSGID_LIM_ERR_EID, 1332
 CFE_SB_PART_SUB_PKT_EID, 1332
 CFE_SB_PIPE_ADDED_EID, 1332
 CFE_SB_PIPE_DELETED_EID, 1333
 CFE_SB_Q_FULL_ERR_EID, 1333
 CFE_SB_Q_RD_ERR_EID, 1333
 CFE_SB_Q_WR_ERR_EID, 1334
 CFE_SB_RCV_BAD_ARG_EID, 1334
 CFE_SB_SEND_BAD_ARG_EID, 1334
 CFE_SB_SEND_INV_MSGID_EID, 1335
 CFE_SB_SEND_NO_SUBS_EID, 1335
 CFE_SB_SETPIPEOPTS_EID, 1335
 CFE_SB_SETPIPEOPTS_ID_ERR_EID, 1336
 CFE_SB_SETPIPEOPTS_OWNER_ERR_EID, 1336
 CFE_SB SND RTG_EID, 1336
 CFE_SB SND RTG_ERR1_EID, 1337
 CFE_SB SND STATS_EID, 1337
 CFE_SB SUB ARG_ERR_EID, 1337
 CFE_SB SUB INV CALLER_EID, 1338
 CFE_SB SUB INV PIPE_EID, 1338
 CFE_SB_SUBSCRIPTION_RCVD_EID, 1338
 CFE_SB_SUBSCRIPTION_REMOVED_EID, 1339
 CFE_SB_SUBSCRIPTION_RPT_EID, 1339
 CFE_SB_UNSUB_ARG_ERR_EID, 1339
 CFE_SB_UNSUB_INV_CALLER_EID, 1340
 CFE_SB_UNSUB_INV_PIPE_EID, 1340
 CFE_SB_UNSUB_NO_SUBS_EID, 1340
 cfe_sb_extern_typedefs.h
 CFE_SB_MsgId_Atom_t, 1175
 CFE_SB_Pipeld_t, 1175
 CFE_SB_QosPriority, 1176
 CFE_SB_QosPriority_Enum_t, 1175
 CFE_SB_QosReliability, 1176
 CFE_SB_QosReliability_Enum_t, 1175
 CFE_SB_Routeld_Atom_t, 1176
 CFE_SB_SUB_ENTRIES_PER_PKT, 1174
 cfe_sb_msg.h
 CFE_SB_AllSubscriptionsTlm_Payload_t, 1354
 CFE_SB_AllSubscriptionsTlm_t, 1354

CFE_SB_DISABLE_ROUTE_CC, 1343
CFE_SB_DISABLE_SUB_REPORTING_CC, 1343
CFE_SB_DisableRouteCmd_t, 1354
CFE_SB_DisableSubReportingCmd_t, 1354
CFE_SB_ENABLE_ROUTE_CC, 1344
CFE_SB_ENABLE_SUB_REPORTING_CC, 1345
CFE_SB_EnableRouteCmd_t, 1354
CFE_SB_EnableSubReportingCmd_t, 1354
CFE_SB_HousekeepingTlm_Payload_t, 1355
CFE_SB_HousekeepingTlm_t, 1355
CFE_SB_MsgMapFileEntry_t, 1355
CFE_SB_NOOP_CC, 1346
CFE_SB_NoopCmd_t, 1355
CFE_SB_PipeDepthStats_t, 1355
CFE_SB_PipeInfoEntry_t, 1355
CFE_SB_RESET_COUNTERS_CC, 1347
CFE_SB_ResetCountersCmd_t, 1356
CFE_SB_RouteCmd_Payload_t, 1356
CFE_SB_RouteCmd_t, 1356
CFE_SB_RoutingFileEntry_t, 1356
CFE_SB_SEND_PREV_SUBS_CC, 1348
CFE_SB_SEND_SB_STATS_CC, 1349
CFE_SB_SendPrevSubsCmd_t, 1356
CFE_SB_SendSbStatsCmd_t, 1357
CFE_SB_SingleSubscriptionTlm_Payload_t, 1357
CFE_SB_SingleSubscriptionTlm_t, 1357
CFE_SB_StatsTlm_Payload_t, 1357
CFE_SB_StatsTlm_t, 1357
CFE_SB_SubEntries_t, 1358
CFE_SB_WRITE_MAP_INFO_CC, 1350
CFE_SB_WRITE_PIPE_INFO_CC, 1351
CFE_SB_WRITE_ROUTING_INFO_CC, 1352
CFE_SB_WriteFileInfoCmd_Payload_t, 1358
CFE_SB_WriteFileInfoCmd_t, 1358
CFE_SB_WriteMapInfoCmd_t, 1358
CFE_SB_WritePipeInfoCmd_t, 1358
CFE_SB_WriteRoutingInfoCmd_t, 1358
cfe_tbl_api_typedefs.h
CFE_TBL_BAD_TABLE_HANDLE, 1180
CFE_TBL_CallbackFuncPtr_t, 1180
CFE_TBL_Handle_t, 1180
CFE_TBL_Info_t, 1180
CFE_TBL_MAX_FULL_NAME_LEN, 1180
CFE_TBL_SrcEnum, 1181
CFE_TBL_SrcEnum_t, 1181
cfe_tbl_events.h
CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID, 1361
CFE_TBL_ACTIVATE_ERR_EID, 1362
CFE_TBL_ASSUMED_VALID_INF_EID, 1362
CFE_TBL_CC1_ERR_EID, 1362
CFE_TBL_CDS_DELETE_ERR_EID, 1363
CFE_TBL_CDS_DELETED_INFO_EID, 1363
CFE_TBL_CDS_NOT_FOUND_ERR_EID, 1363
CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID, 1364
CFE_TBL_CREATING_DUMP_FILE_ERR_EID, 1364
CFE_TBL_DUMP_PENDING_ERR_EID, 1364
CFE_TBL_FAIL_HK_SEND_ERR_EID, 1365
CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID, 1365
CFE_TBL_FILE_ACCESS_ERR_EID, 1365
CFE_TBL_FILE_INCOMPLETE_ERR_EID, 1366
CFE_TBL_FILE_LOADED_INF_EID, 1366
CFE_TBL_FILE_STD_HDR_ERR_EID, 1366
CFE_TBL_FILE_SUBTYPE_ERR_EID, 1367
CFE_TBL_FILE_TBL_HDR_ERR_EID, 1367
CFE_TBL_FILE_TOO_BIG_ERR_EID, 1367
CFE_TBL_FILE_TYPE_ERR_EID, 1368
CFE_TBL_HANDLE_ACCESS_ERR_EID, 1368
CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID, 1368
CFE_TBL_IN_REGISTRY_ERR_EID, 1369
CFE_TBL_INIT_INF_EID, 1369
CFE_TBL_INTERNAL_ERROR_ERR_EID, 1369
CFE_TBL_LEN_ERR_EID, 1370
CFE_TBL_LOAD_ABORT_ERR_EID, 1370
CFE_TBL_LOAD_ABORT_INF_EID, 1370
CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID, 1371
CFE_TBL_LOAD_FILENAME_LONG_ERR_EID, 1371
CFE_TBL_LOAD_IN_PROGRESS_ERR_EID, 1371
CFE_TBL_LOAD_PEND_REQ_INF_EID, 1372
CFE_TBL_LOAD_SUCCESS_INF_EID, 1372
CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID, 1372
CFE_TBL_LOAD_TYPE_ERR_EID, 1373
CFE_TBL_LOAD_VAL_ERR_EID, 1373
CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID, 1373
CFE_TBL_LOADING_PENDING_ERR_EID, 1374
CFE_TBL_MID_ERR_EID, 1374
CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID, 1374
CFE_TBL_NO_SUCH_TABLE_ERR_EID, 1375
CFE_TBL_NO_WORK_BUFFERS_ERR_EID, 1375
CFE_TBL_NOOP_INF_EID, 1375
CFE_TBL_NOT_CRITICAL_TBL_ERR_EID, 1376
CFE_TBL_NOT_IN_CRIT_REG_ERR_EID, 1376
CFE_TBL_OVERWRITE_DUMP_INF_EID, 1376
CFE_TBL_OVERWRITE_REG_DUMP_INF_EID, 1377
CFE_TBL_PARTIAL_LOAD_ERR_EID, 1377
CFE_TBL_PROCESSOR_ID_ERR_EID, 1377
CFE_TBL_REGISTER_ERR_EID, 1378
CFE_TBL_RESET_INF_EID, 1378
CFE_TBL_SHARE_ERR_EID, 1378
CFE_TBL_SPACECRAFT_ID_ERR_EID, 1379
CFE_TBL_TLM_REG_CMD_INF_EID, 1379

CFE_TBL_TOO_MANY_DUMPS_ERR_EID, 1379
 CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID, 1380
 CFE_TBL_UNREGISTER_ERR_EID, 1380
 CFE_TBL_UNVALIDATED_ERR_EID, 1380
 CFE_TBL_UPDATE_ERR_EID, 1381
 CFE_TBL_UPDATE_SUCCESS_INF_EID, 1381
 CFE_TBL_VAL_REQ_MADE_INF_EID, 1381
 CFE_TBL_VALIDATION_ERR_EID, 1382
 CFE_TBL_VALIDATION_INF_EID, 1382
 CFE_TBL_WRITE_CFE_HDR_ERR_EID, 1382
 CFE_TBL_WRITE_DUMP_INF_EID, 1383
 CFE_TBL_WRITE_REG_DUMP_INF_EID, 1383
 CFE_TBL_WRITE_TBL_HDR_ERR_EID, 1383
 CFE_TBL_WRITE_TBL_IMG_ERR_EID, 1384
 CFE_TBL_WRITE_TBL_REG_ERR_EID, 1384
 CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID, 1384
cfe_tbl_extern_typedefs.h
 CFE_TBL_BufferSelect, 1182
 CFE_TBL_BufferSelect_Enum_t, 1182
 CFE_TBL_File_Hdr_t, 1182
cfe_tbl_filedef.h
 CFE_TBL_FILEDEF, 1184
 CFE_TBL_FileDef_t, 1184
cfe_tbl_msg.h
 CFE_TBL_ABORT_LOAD_CC, 1387
 CFE_TBL_ACTIVATE_CC, 1388
 CFE_TBL_AbortLoadCmd_Payload_t, 1398
 CFE_TBL_AbortLoadCmd_t, 1398
 CFE_TBL_ActivateCmd_Payload_t, 1398
 CFE_TBL_ActivateCmd_t, 1398
 CFE_TBL_DELETE_CDS_CC, 1389
 CFE_TBL_DUMP_CC, 1390
 CFE_TBL_DUMP_REGISTRY_CC, 1391
 CFE_TBL_DelCDSCmd_Payload_t, 1398
 CFE_TBL_DeleteCDSCmd_t, 1398
 CFE_TBL_DumpCmd_Payload_t, 1399
 CFE_TBL_DumpCmd_t, 1399
 CFE_TBL_DumpRegistryCmd_Payload_t, 1399
 CFE_TBL_DumpRegistryCmd_t, 1399
 CFE_TBL_HousekeepingTlm_Payload_t, 1399
 CFE_TBL_HousekeepingTlm_t, 1399
 CFE_TBL_LOAD_CC, 1392
 CFE_TBL_LoadCmd_Payload_t, 1400
 CFE_TBL_LoadCmd_t, 1400
 CFE_TBL_NOOP_CC, 1393
 CFE_TBL_NoArgsCmd_t, 1400
 CFE_TBL_NoopCmd_t, 1400
 CFE_TBL_NotifyCmd_Payload_t, 1400
 CFE_TBL_NotifyCmd_t, 1401
 CFE_TBL_RESET_COUNTERS_CC, 1394
 CFE_TBL_ResetCountersCmd_t, 1401
 CFE_TBL_SEND_REGISTRY_CC, 1395
 CFE_TBL_SendRegistryCmd_Payload_t, 1401
 CFE_TBL_SendRegistryCmd_t, 1401
 CFE_TBL_TableRegistryTlm_t, 1401
 CFE_TBL_TblRegPacket_Payload_t, 1401
 CFE_TBL_VALIDATE_CC, 1396
 CFE_TBL_ValidateCmd_Payload_t, 1402
 CFE_TBL_ValidateCmd_t, 1402
cfe_time.h
 CFE_TIME_Copy, 1186
cfe_time_api_typedefs.h
 CFE_TIME_Compare, 1188
 CFE_TIME_Compare_t, 1188
 CFE_TIME_PRINTED_STRING_SIZE, 1187
 CFE_TIME_SynchCallbackPtr_t, 1188
cfe_time_events.h
 CFE_TIME_1HZ_CFG_EID, 1404
 CFE_TIME_1HZ_EID, 1404
 CFE_TIME_CC_ERR_EID, 1404
 CFE_TIME_DELAY_CFG_EID, 1405
 CFE_TIME_DELAY_EID, 1405
 CFE_TIME_DELAY_ERR_EID, 1405
 CFE_TIME_DELTA_CFG_EID, 1406
 CFE_TIME_DELTA_EID, 1406
 CFE_TIME_DELTA_ERR_EID, 1406
 CFE_TIME_DIAG_EID, 1407
 CFE_TIME_FLY_OFF_EID, 1407
 CFE_TIME_FLY_ON_EID, 1407
 CFE_TIME_ID_ERR_EID, 1408
 CFE_TIME_INIT_EID, 1408
 CFE_TIME_LEAPS_CFG_EID, 1408
 CFE_TIME_LEAPS_EID, 1409
 CFE_TIME_LEN_ERR_EID, 1409
 CFE_TIME_MET_CFG_EID, 1409
 CFE_TIME_MET_EID, 1410
 CFE_TIME_MET_ERR_EID, 1410
 CFE_TIME_NOOP_EID, 1410
 CFE_TIME_RESET_EID, 1411
 CFE_TIME_SIGNAL_CFG_EID, 1411
 CFE_TIME_SIGNAL_EID, 1411
 CFE_TIME_SIGNAL_ERR_EID, 1412
 CFE_TIME_SOURCE_CFG_EID, 1412
 CFE_TIME_SOURCE_EID, 1412
 CFE_TIME_SOURCE_ERR_EID, 1413
 CFE_TIME_STATE_EID, 1413
 CFE_TIME_STATE_ERR_EID, 1413
 CFE_TIME_STCF_CFG_EID, 1414
 CFE_TIME_STCF_EID, 1414
 CFE_TIME_STCF_ERR_EID, 1414
 CFE_TIME_TIME_CFG_EID, 1415
 CFE_TIME_TIME_EID, 1415
 CFE_TIME_TIME_ERR_EID, 1415
cfe_time_extern_typedefs.h
 CFE_TIME_AdjustDirection, 1192
 CFE_TIME_AdjustDirection_Enum_t, 1190
 CFE_TIME_ClockState, 1193

CFE_TIME_ClockState_Enum_t, 1190
CFE_TIME_FlagBit, 1193
CFE_TIME_FlagBit_Enum_t, 1191
CFE_TIME_FlywheelState, 1194
CFE_TIME_FlywheelState_Enum_t, 1191
CFE_TIME_SetState, 1194
CFE_TIME_SetState_Enum_t, 1191
CFE_TIME_SourceSelect, 1194
CFE_TIME_SourceSelect_Enum_t, 1192
CFE_TIME_SysTime_t, 1192
CFE_TIME_ToneSignalSelect, 1194
CFE_TIME_ToneSignalSelect_Enum_t, 1192
cfe_time_msg.h
 CFE_TIME_1HzCmd_t, 1436
 CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, 1419
 CFE_TIME_ADD_ADJUST_CC, 1420
 CFE_TIME_ADD_DELAY_CC, 1421
 CFE_TIME_Add1HZAdjustmentCmd_t, 1437
 CFE_TIME_AddAdjustCmd_t, 1437
 CFE_TIME_AddDelayCmd_t, 1437
 CFE_TIME_DiagnosticTlm_Payload_t, 1437
 CFE_TIME_DiagnosticTlm_t, 1437
 CFE_TIME_FakeToneCmd_t, 1437
 CFE_TIME_HousekeepingTlm_Payload_t, 1438
 CFE_TIME_HousekeepingTlm_t, 1438
 CFE_TIME_LeapsCmd_Payload_t, 1438
 CFE_TIME_NOOP_CC, 1422
 CFE_TIME_NoArgsCmd_t, 1438
 CFE_TIME_NoopCmd_t, 1438
 CFE_TIME_OneHzAdjustmentCmd_Payload_t, 1438
 CFE_TIME_OneHzAdjustmentCmd_t, 1439
 CFE_TIME_RESET_COUNTERS_CC, 1423
 CFE_TIME_ResetCountersCmd_t, 1439
 CFE_TIME_SEND_DIAGNOSTIC_TLM_CC, 1424
 CFE_TIME_SET_LEAP_SECONDS_CC, 1425
 CFE_TIME_SET_MET_CC, 1426
 CFE_TIME_SET_SIGNAL_CC, 1427
 CFE_TIME_SET_SOURCE_CC, 1428
 CFE_TIME_SET_STATE_CC, 1429
 CFE_TIME_SET_STCF_CC, 1431
 CFE_TIME_SET_TIME_CC, 1432
 CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, 1433
 CFE_TIME_SUB_ADJUST_CC, 1434
 CFE_TIME_SUB_DELAY_CC, 1435
 CFE_TIME_SendDiagnosticCmd_t, 1439
 CFE_TIME_SetLeapSecondsCmd_t, 1439
 CFE_TIME_SetMETCmd_t, 1439
 CFE_TIME_SetSTCFCmd_t, 1440
 CFE_TIME_SetSignalCmd_t, 1439
 CFE_TIME_SetSourceCmd_t, 1440
 CFE_TIME_SetStateCmd_t, 1440
 CFE_TIME_SetTimeCmd_t, 1440
 CFE_TIME_SignalCmd_Payload_t, 1440
 CFE_TIME_SourceCmd_Payload_t, 1440
CFE_TIME_StateCmd_Payload_t, 1441
CFE_TIME_Sub1HZAdjustmentCmd_t, 1441
CFE_TIME_SubAdjustCmd_t, 1441
CFE_TIME_SubDelayCmd_t, 1441
CFE_TIME_TimeCmd_Payload_t, 1441
CFE_TIME_TimeCmd_t, 1441
CFE_TIME_ToneDataCmd_Payload_t, 1442
CFE_TIME_ToneDataCmd_t, 1442
CFE_TIME_ToneSignalCmd_t, 1442
cfe_version.h
 CFE_BUILD_BASELINE, 1196
 CFE_BUILD_NUMBER, 1196
 CFE_MAJOR_VERSION, 1196
 CFE_MINOR_VERSION, 1196
 CFE_MISSION_REV, 1196
 CFE_REVISION, 1196
 CFE_SRC_VERSION, 1197
 CFE_STR_HELPER, 1197
 CFE_STR, 1197
 CFE_VERSION_STRING, 1197
CheckErrCtr
 CFE_ES_MemPoolStats, 618
ClockFlyState
 CFE_TIME_DiagnosticTlm_Payload, 761
ClockSetState
 CFE_TIME_DiagnosticTlm_Payload, 761
ClockSignal
 CFE_TIME_DiagnosticTlm_Payload, 761
ClockSource
 CFE_TIME_DiagnosticTlm_Payload, 761
ClockState
 CFE_TIME_StateCmd_Payload, 785
ClockStateAPI
 CFE_TIME_DiagnosticTlm_Payload, 762
 CFE_TIME_HousekeepingTlm_Payload, 772
ClockStateFlags
 CFE_TIME_DiagnosticTlm_Payload, 762
 CFE_TIME_HousekeepingTlm_Payload, 772
CmdCount
 HS_AppData_t, 796
 HS_HkPacket_t, 813
CmdErrCount
 HS_AppData_t, 796
 HS_HkPacket_t, 813
CmdHeader
 HS_NoArgsCmd_t, 820
 HS_SetMaxResetsCmd_t, 820
 HS_SetUtilDiagCmd_t, 822
 HS_SetUtilParamsCmd_t, 823
CmdPipe
 HS_AppData_t, 797
code_address
 OS_module_address_t, 835
code_size

OS_module_address_t, 835
CodeAddress
 CFE_ES_AppInfo, 588
CodeSize
 CFE_ES_AppInfo, 588
CommandCounter
 CFE_ES_HousekeepingTlm_Payload, 608
 CFE_EVS_HousekeepingTlm_Payload, 663
 CFE_SB_HousekeepingTlm_Payload, 689
 CFE_TBL_HousekeepingTlm_Payload, 734
 CFE_TIME_HousekeepingTlm_Payload, 772
CommandErrorCounter
 CFE_ES_HousekeepingTlm_Payload, 608
 CFE_EVS_HousekeepingTlm_Payload, 663
 CFE_SB_HousekeepingTlm_Payload, 689
 CFE_TBL_HousekeepingTlm_Payload, 734
 CFE_TIME_HousekeepingTlm_Payload, 773
CommandHeader
 CFE_ES_AppNameCmd, 593
 CFE_ES_DeleteCDSCmd, 599
 CFE_ES_DumpCDSRegistryCmd, 601
 CFE_ES_FileNameCmd, 602
 CFE_ES_NoArgsCmd, 621
 CFE_ES_OverWriteSysLogCmd, 623
 CFE_ES_ReloadAppCmd, 627
 CFE_ES_RestartCmd, 628
 CFE_ES_SendMemPoolStatsCmd, 630
 CFE_ES_SetMaxPRCountCmd, 632
 CFE_ES_SetPerfFilterMaskCmd, 634
 CFE_ES_SetPerfTriggerMaskCmd, 636
 CFE_ES_StartApp, 638
 CFE_ES_StartPerfDataCmd, 642
 CFE_ES_StopPerfDataCmd, 644
 CFE_EVS_AppNameBitMaskCmd, 648
 CFE_EVS_AppNameCmd, 650
 CFE_EVS_AppNameEventIDCmd, 652
 CFE_EVS_AppNameEventIDMaskCmd, 654
 CFE_EVS_BitMaskCmd, 659
 CFE_EVS_NoArgsCmd, 670
 CFE_EVS_SetEventFormatModeCmd, 674
 CFE_EVS_SetLogModeCmd, 676
 CFE_EVS_WriteAppDataFileCmd, 678
 CFE_EVS_WriteLogDataFileCmd, 679
 CFE_SB_RouteCmd, 703
 CFE_SB_WriteFileInfoCmd, 717
 CFE_TBL_AbortLoadCmd, 719
 CFE_TBL_ActivateCmd, 721
 CFE_TBL_DeleteCDSCmd, 723
 CFE_TBL_DumpCmd, 724
 CFE_TBL_DumpRegistryCmd, 727
 CFE_TBL_LoadCmd, 742
 CFE_TBL_NoArgsCmd, 744
 CFE_TBL_NotifyCmd, 745
 CFE_TBL_SendRegistryCmd, 747

CFE_TBL_ValidateCmd, 755
CFE_TIME_NoArgsCmd, 777
CFE_TIME_OneHzAdjustmentCmd, 777
CFE_TIME_SetLeapSecondsCmd, 779
CFE_TIME_SetSignalCmd, 780
CFE_TIME_SetSourceCmd, 781
CFE_TIME_SetStateCmd, 782
CFE_TIME_TimeCmd, 787
CFE_TIME_ToneDataCmd, 788
common_types.h
 EXTENSION, 1444
 CompileTimeAssert, 1444, 1449, 1450
 cpuaddr, 1445
 cpudiff, 1446
 cpusize, 1446
 int16, 1446
 int32, 1446
 int64, 1446
 int8, 1446
 intptr, 1447
 OS_ArgCallback_t, 1447
 OS_PRINTF, 1444
 OS_USED, 1444
 OSAL_BLOCKCOUNT_C, 1444
 OSAL_INDEX_C, 1445
 OSAL_OBJTYPE_C, 1445
 OSAL_SIZE_C, 1445
 OSAL_STATUS_C, 1445
 osal_blockcount_t, 1447
 osal_id_t, 1447
 osal_index_t, 1447
 osal_objtype_t, 1448
 osal_status_t, 1448
 uint16, 1448
 uint32, 1448
 uint64, 1448
 uint8, 1449
 CompileTimeAssert
 common_types.h, 1444, 1449, 1450
ContentType
 CFE_FS_Header, 682
Cooldown
 HS_MATEEntry_t, 817
cpu1_msgids.h
 CFE_ES_APP_TLM_MID, 1002
 CFE_ES_CMD_MID, 1002
 CFE_ES_HK_TLM_MID, 1002
 CFE_ES_MEMSTATS_TLM_MID, 1003
 CFE_ES_SEND_HK_MID, 1003
 CFE_EVS_CMD_MID, 1003
 CFE_EVS_HK_TLM_MID, 1003
 CFE_EVS_LONG_EVENT_MSG_MID, 1003
 CFE_EVS_SEND_HK_MID, 1003
 CFE_EVS_SHORT_EVENT_MSG_MID, 1004

CFE_PLATFORM_CMD_MID_BASE_GLOB, 1004
CFE_PLATFORM_CMD_MID_BASE, 1004
CFE_PLATFORM_TLM_MID_BASE, 1004
CFE_SB_ALLSUBS_TLM_MID, 1005
CFE_SB_CMD_MID, 1005
CFE_SB_HK_TLM_MID, 1005
CFE_SB_ONESUB_TLM_MID, 1005
CFE_SB_SEND_HK_MID, 1005
CFE_SB_STATS_TLM_MID, 1006
CFE_SB_SUB_RPT_CTRL_MID, 1006
CFE_TBL_CMD_MID, 1006
CFE_TBL_HK_TLM_MID, 1006
CFE_TBL_REG_TLM_MID, 1006
CFE_TBL_SEND_HK_MID, 1006
CFE_TEST_CMD_MID, 1007
CFE_TEST_HK_TLM_MID, 1007
CFE_TIME_1HZ_CMD_MID, 1007
CFE_TIME_CMD_MID, 1007
CFE_TIME_DATA_CMD_MID, 1007
CFE_TIME_DIAG_TLM_MID, 1007
CFE_TIME_HK_TLM_MID, 1008
CFE_TIME_SEND_CMD_MID, 1008
CFE_TIME_SEND_HK_MID, 1008
CFE_TIME_TONE_CMD_MID, 1008
cpu1_platform_cfg.h
CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, 1012
CFE_PLATFORM_ENDIAN, 1013
CFE_PLATFORM_ES_APP_KILL_TIMEOUT, 1013
CFE_PLATFORM_ES_APP_SCAN_RATE, 1014
CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, 1014
CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, 1015
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01, 1015
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02, 1015
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08, 1016
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14, 1017
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15, 1018
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16, 1018
CFE_PLATFORM_ES_CDS_SIZE, 1018
CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE, 1018
CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMPF_FILE, 1019
CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE, 1019
CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME, 1019
CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MODE, 1020
CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE, 1020
CFE_PLATFORM_ES_DEFAULT_STACK_SIZE, 1021
CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE, 1021
CFE_PLATFORM_ES_DEFAULT_TASK_LOGFILE, 1022
CFE_PLATFORM_ES_ER_LOG_ENTRIES, 1022
CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE, 1023
CFE_PLATFORM_ES_MAX_APPLICATIONS, 1023
CFE_PLATFORM_ES_MAX_BLOCK_SIZE, 1023
CFE_PLATFORM_ES_MAX_GEN_COUNTERS, 1024
CFE_PLATFORM_ES_MAX_LIBRARIES, 1024
CFE_PLATFORM_ES_MAX_MEMORY_POOLS, 1024
CFE_PLATFORM_ES_MAX_PROCESSOR_RESOURCES, 1025
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01, 1025
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02, 1026
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03, 1026
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04, 1026
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05, 1026
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06,

1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07,
 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08,
 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09,
 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10,
 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11,
 1027
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12,
 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13,
 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14,
 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15,
 1028
 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16,
 1028
 CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE←
 _MIN, 1028
 CFE_PLATFORM_ES_NONVOL_DISK_MOUNT←
 _STRING, 1029
 CFE_PLATFORM_ES_NONVOL_STARTUP_FILE,
 1029
 CFE_PLATFORM_ES_OBJECT_TABLE_SIZE,
 1029
 CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY,
 1030
 CFE_PLATFORM_ES_PERF_CHILD_PRIORITY,
 1030
 CFE_PLATFORM_ES_PERF_CHILD_STACK_SI←
 ZE, 1030
 CFE_PLATFORM_ES_PERF_DATA_BUFFER_SI←
 ZE, 1031
 CFE_PLATFORM_ES_PERF_ENTRIES_BTWN←
 DLYS, 1031
 CFE_PLATFORM_ES_PERF_FILTMASK_ALL,
 1031
 CFE_PLATFORM_ES_PERF_FILTMASK_INIT,
 1032
 CFE_PLATFORM_ES_PERF_FILTMASK_NONE,
 1032
 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL,
 1032
 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT,
 1033
 CFE_PLATFORM_ES_PERF_TRIGMASK_NONE,
 1033
 CFE_PLATFORM_ES_POOL_MAX_BUCKETS,
 1033
 CFE_PLATFORM_ES_RAM_DISK_MOUNT_STR←

ING, 1034
 CFE_PLATFORM_ES_RAM_DISK_NUM_SECTO←
 RS, 1034
 CFE_PLATFORM_ES_RAM_DISK_PERCENT_R←
 ESERVED, 1034
 CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE,
 1035
 CFE_PLATFORM_ES_RESET_AREA_SIZE, 1035
 CFE_PLATFORM_ES_START_TASK_PRIORITY,
 1036
 CFE_PLATFORM_ES_START_TASK_STACK_SI←
 ZE, 1036
 CFE_PLATFORM_ES_STARTUP_SCRIPT_TIME←
 OUT_MSEC, 1037
 CFE_PLATFORM_ES_STARTUP_SYNC_POLL←
 MSEC, 1037
 CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, 1038
 CFE_PLATFORM_ES_USER_RESERVED_SIZE,
 1038
 CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE,
 1039
 CFE_PLATFORM_EVS_DEFAULT_APP_DATA←
 FILE, 1039
 CFE_PLATFORM_EVS_DEFAULT_LOG_FILE,
 1040
 CFE_PLATFORM_EVS_DEFAULT_LOG_MODE,
 1040
 CFE_PLATFORM_EVS_DEFAULT_MSG_FORM←
 AT_MODE, 1040
 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG,
 1041
 CFE_PLATFORM_EVS_LOG_MAX, 1041
 CFE_PLATFORM_EVS_MAX_EVENT_FILTERS,
 1041
 CFE_PLATFORM_EVS_PORT_DEFAULT, 1042
 CFE_PLATFORM_EVS_START_TASK_PRIORITY,
 1042
 CFE_PLATFORM_EVS_START_TASK_STACK←
 SIZE, 1042
 CFE_PLATFORM_SB_BUF_MEMORY_BYTES,
 1043
 CFE_PLATFORM_SB_DEFAULT_MAP_FILENA←
 ME, 1043
 CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT,
 1044
 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENA←
 ME, 1044
 CFE_PLATFORM_SB_DEFAULT_ROUTING_FIL←
 ENAME, 1045
 CFE_PLATFORM_SB_FILTER_MASK1, 1045
 CFE_PLATFORM_SB_FILTER_MASK2, 1045
 CFE_PLATFORM_SB_FILTER_MASK3, 1046
 CFE_PLATFORM_SB_FILTER_MASK4, 1046
 CFE_PLATFORM_SB_FILTER_MASK5, 1046

CFE_PLATFORM_SB_FILTER_MASK6, 1046
CFE_PLATFORM_SB_FILTER_MASK7, 1046
CFE_PLATFORM_SB_FILTER_MASK8, 1046
CFE_PLATFORM_SB_FILTERED_EVENT1, 1047
CFE_PLATFORM_SB_FILTERED_EVENT2, 1047
CFE_PLATFORM_SB_FILTERED_EVENT3, 1047
CFE_PLATFORM_SB_FILTERED_EVENT4, 1047
CFE_PLATFORM_SB_FILTERED_EVENT5, 1047
CFE_PLATFORM_SB_FILTERED_EVENT6, 1048
CFE_PLATFORM_SB_FILTERED_EVENT7, 1048
CFE_PLATFORM_SB_FILTERED_EVENT8, 1048
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID,
 1048
CFE_PLATFORM_SB_MAX_BLOCK_SIZE, 1048
CFE_PLATFORM_SB_MAX_DEST_PER_PKT,
 1049
CFE_PLATFORM_SB_MAX_MSG_IDS, 1049
CFE_PLATFORM_SB_MAX_PIPES, 1049
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01,
 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02,
 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03,
 1050
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09,
 1051
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15,
 1052
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16,
 1053
CFE_PLATFORM_SB_START_TASK_PRIORITY,
 1053
CFE_PLATFORM_SB_START_TASK_STACK_SI←
ZE, 1053
CFE_PLATFORM_TBL_BUF_MEMORY_BYTES,
 1053
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_↪
FILE, 1054
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES,
 1054
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE,
 1054
CFE_PLATFORM_TBL_MAX_NUM_HANDLES,
 1055
CFE_PLATFORM_TBL_MAX_NUM_TABLES, 1055
CFE_PLATFORM_TBL_MAX_NUM_VALIFICATIONS,
 1055
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_↪
LOADS, 1056
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE,
 1056
CFE_PLATFORM_TBL_START_TASK_PRIORITY,
 1057
CFE_PLATFORM_TBL_START_TASK_STACK_↪
SIZE, 1057
CFE_PLATFORM_TBL_U32FROM4CHARS, 1057
CFE_PLATFORM_TBL_VALID_PRID_1, 1058
CFE_PLATFORM_TBL_VALID_PRID_2, 1058
CFE_PLATFORM_TBL_VALID_PRID_3, 1058
CFE_PLATFORM_TBL_VALID_PRID_4, 1058
CFE_PLATFORM_TBL_VALID_PRID_COUNT,
 1059
CFE_PLATFORM_TBL_VALID_SCID_1, 1059
CFE_PLATFORM_TBL_VALID_SCID_2, 1059
CFE_PLATFORM_TBL_VALID_SCID_COUNT,
 1060
CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY,
 1060
CFE_PLATFORM_TIME_1HZ_TASK_STACK_SI←
ZE, 1060
CFE_PLATFORM_TIME_CFG_CLIENT, 1060
CFE_PLATFORM_TIME_CFG_LATCH_FLY, 1060
CFE_PLATFORM_TIME_CFG_SERVER, 1061
CFE_PLATFORM_TIME_CFG_SIGNAL, 1061
CFE_PLATFORM_TIME_CFG_SOURCE, 1061
CFE_PLATFORM_TIME_CFG_SRC_GPS, 1062
CFE_PLATFORM_TIME_CFG_SRC_MET, 1062
CFE_PLATFORM_TIME_CFG_SRC_TIME, 1062
CFE_PLATFORM_TIME_CFG_START_FLY, 1063
CFE_PLATFORM_TIME_CFG_TONE_LIMIT, 1063
CFE_PLATFORM_TIME_CFG_VIRTUAL, 1063
CFE_PLATFORM_TIME_MAX_DELTA_SECS, 1064
CFE_PLATFORM_TIME_MAX_DELTA_SUBS, 1064
CFE_PLATFORM_TIME_MAX_LOCAL_SECS,
 1065
CFE_PLATFORM_TIME_MAX_LOCAL_SUBS,
 1065
CFE_PLATFORM_TIME_START_TASK_PRIORI↪

TY, 1065
 CFE_PLATFORM_TIME_START_TASK_STACK_SIZE, 1065
 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY, 1066
 CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE, 1066
 cpuaddr
 common_types.h, 1445
 cpudiff
 common_types.h, 1446
 cupsize
 common_types.h, 1446
 Crc
 CFE_TBL_Info, 739
 CFE_TBL_TblRegPacket_Payload, 751
 CreatePipeErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 689
 creator
 OS_bin_sem_prop_t, 826
 OS_count_sem_prop_t, 827
 OS_mut_sem_prop_t, 838
 OS_queue_prop_t, 839
 OS_socket_prop_t, 842
 OS_task_prop_t, 845
 OS_timebase_prop_t, 847
 OS_timer_prop_t, 849
 Critical
 CFE_TBL_Info, 739
 CFE_TBL_TblRegPacket_Payload, 751
 CurrentAlivenessState
 HS_AppData_t, 797
 HS_HkPacket_t, 813
 CurrentAppMonState
 HS_AppData_t, 797
 HS_HkPacket_t, 813
 CurrentCPUHogState
 HS_AppData_t, 798
 HS_HkPacket_t, 814
 CurrentCPUHoggingTime
 HS_AppData_t, 797
 CurrentCPUUtilIndex
 HS_AppData_t, 798
 CurrentEventMonState
 HS_AppData_t, 798
 HS_HkPacket_t, 814
 CurrentLatch
 CFE_TIME_DiagnosticTlm_Payload, 762
 CurrentMET
 CFE_TIME_DiagnosticTlm_Payload, 762
 CurrentQueueDepth
 CFE_SB_PipeDepthStats, 697
 CFE_SB_PipeInfoEntry, 699
 CurrentTAI
 CFE_TIME_DiagnosticTlm_Payload, 763
 CurrentUTC
 CFE_TIME_DiagnosticTlm_Payload, 763
 CycleCount
 HS_AMTEEntry_t, 791
 data_address
 OS_module_address_t, 835
 data_size
 OS_module_address_t, 835
 DataAddress
 CFE_ES_AppInfo, 588
 DataFileName
 CFE_ES_StopPerfCmd_Payload, 643
 DataSize
 CFE_ES_AppInfo, 589
 DataStoreStatus
 CFE_TIME_DiagnosticTlm_Payload, 763
 DelayDirection
 CFE_TIME_DiagnosticTlm_Payload, 763
 Description
 CFE_FS_FileWriteMetaData, 680
 CFE_FS_Header, 683
 CFE_TBL_FileDef, 730
 Div
 HS_SetUtilParamsCmd_t, 823
 DoubleBuffered
 CFE_TBL_Info, 740
 CFE_TBL_TblRegPacket_Payload, 751
 DumpFilename
 CFE_ES_DumpCDSRegistryCmd_Payload, 602
 CFE_TBL_DumpCmd_Payload, 726
 CFE_TBL_DumpRegistryCmd_Payload, 728
 DumpOnly
 CFE_TBL_Info, 740
 CFE_TBL_TblRegPacket_Payload, 751
 DuplicateSubscriptionsCounter
 CFE_SB_HousekeepingTlm_Payload, 689
 EMTableHandle
 HS_AppData_t, 798
 EMTablePtr
 HS_AppData_t, 799
 ERLogEntries
 CFE_ES_HousekeepingTlm_Payload, 608
 ERLogIndex
 CFE_ES_HousekeepingTlm_Payload, 609
 EnableState
 HS_MATEEntry_t, 817
 Entries
 CFE_SB_AllSubscriptionsTlm_Payload, 686
 Entry
 CFE_SB_AllSubscriptionsTlm_Payload, 686
 entry_point
 OS_module_prop_t, 836

EntryPoint
 CFE_ES_AppInfo, 589

EventID
 CFE_EVS_AppNameEventIDCmd_Payload, 653
 CFE_EVS_AppNameEventIDMaskCmd_Payload,
 655
 CFE_EVS_BinFilter, 658
 CFE_EVS_PacketID, 671
 HS_EMTEntry_t, 811

EventMonLoaded
 HS_AppData_t, 799

EventPipe
 HS_AppData_t, 799

EventType
 CFE_EVS_PacketID, 671

EventsMonitoredCount
 HS_AppData_t, 799
 HS_HkPacket_t, 814

ExceptionAction
 CFE_ES_AppInfo, 589
 CFE_ES_StartAppCmd_Payload, 640

ExeCountState
 HS_AppData_t, 800

ExecutionCounter
 CFE_ES_AppInfo, 589
 CFE_ES_TaskInfo, 645

FailedValCounter
 CFE_TBL_HousekeepingTlm_Payload, 734

FileCreateTimeSecs
 CFE_TBL_Info, 740
 CFE_TBL_TblRegPacket_Payload, 752

FileCreateTimeSubSecs
 CFE_TBL_Info, 740
 CFE_TBL_TblRegPacket_Payload, 752

FileModeBits
 os_fstat_t, 832

FileName
 CFE_ES_AppInfo, 590
 CFE_ES_FileNameCmd_Payload, 603
 CFE_FS_FileWriteMetaData, 680
 os_dirent_t, 828

FileSize
 os_fstat_t, 832

FileSubType
 CFE_FS_FileWriteMetaData, 681

FileTime
 os_fstat_t, 832

Filename
 CFE_SB_WriteFileInfoCmd_Payload, 718

filename
 OS_module_prop_t, 837

FilterMask
 CFE_ES_SetPerfFilterMaskCmd_Payload, 635

FilterMaskNum
 CFE_ES_SetPerfFilterMaskCmd_Payload, 635

flags
 OS_module_address_t, 835

Forced2Fly
 CFE_TIME_DiagnosticTlm_Payload, 764

free_blocks
 OS_heap_prop_t, 833

free_bytes
 OS_heap_prop_t, 833

FreeFds
 os_fsinfo_t, 830

FreeVolumes
 os_fsinfo_t, 831

freerun_time
 OS_timebase_prop_t, 848

GetData
 CFE_FS_FileWriteMetaData, 681

GetPipeIdByNameErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 690

HS_ALIVENESS_DEFAULT_STATE
 CFS Health and Safety Platform Configuration, 140

HS_AM_TBL_NULL_ERR_EID
 CFS Health and Safety Event IDs, 162

HS_AMT_ACT_APP_RESTART
 hs_tbldefs.h, 971

HS_AMT_ACT_EVENT
 hs_tbldefs.h, 971

HS_AMT_ACT_LAST_NONMSG
 hs_tbldefs.h, 972

HS_AMT_ACT_MSG
 hs_tbl.h, 969

HS_AMT_ACT_NOACT
 hs_tbldefs.h, 972

HS_AMT_ACT_PROC_RESET
 hs_tbldefs.h, 972

HS_AMT_FILENAME
 CFS Health and Safety Platform Configuration, 141

HS_AMT_LD_ERR_EID
 CFS Health and Safety Event IDs, 163

HS_AMT_REG_ERR_EID
 CFS Health and Safety Event IDs, 163

HS_AMT_TABLENAME
 hs_tbldefs.h, 972

HS_AMTActionsValid
 hs_utils.c, 981
 hs_utils.h, 984

HS_AMTEntry_t, 791
 ActionType, 791
 AppName, 791
 CycleCount, 791
 NullTerm, 792

HS_AMTVAL_ERR_ACT

hs_tbldefs.h, 973
HS_AMTVAL_ERR_EID
 CFS Health and Safety Event IDs, 164
HS_AMTVAL_ERR_NUL
 hs_tbldefs.h, 973
HS_AMTVAL_INF_EID
 CFS Health and Safety Event IDs, 164
HS_AMTVAL_NO_ERR
 hs_tbldefs.h, 973
HS_APP_EXIT_EID
 CFS Health and Safety Event IDs, 165
HS_APP_NAME
 CFS Health and Safety Platform Configuration, 141
HS_APPMAIN_PERF_ID
 CFS Health and Safety Mission Configuration, 135
HS_APPMON_APPNAME_DBG_EID
 CFS Health and Safety Event IDs, 165
HS_APPMON_APPNAME_ERR_EID
 CFS Health and Safety Event IDs, 166
HS_APPMON_DEFAULT_STATE
 CFS Health and Safety Platform Configuration, 141
HS_APPMON_FAIL_ERR_EID
 CFS Health and Safety Event IDs, 166
HS_APPMON_GETADDR_ERR_EID
 CFS Health and Safety Event IDs, 167
HS_APPMON_MSGACTS_ERR_EID
 CFS Health and Safety Event IDs, 167
HS_APPMON_NOT_RESTARTED_ERR_EID
 CFS Health and Safety Event IDs, 168
HS_APPMON_PROC_ERR_EID
 CFS Health and Safety Event IDs, 168
HS_APPMON_RESTART_ERR_EID
 CFS Health and Safety Event IDs, 169
HS_AcquirePointers
 hs_cmds.c, 879
 hs_cmds.h, 899
HS_AppData
 hs_app.c, 864
 hs_app.h, 877
HS_AppData_t, 792
 AMTableHandle, 794
 AMTablePtr, 794
 AlivenessCounter, 794
 AppMonCheckInCountdown, 795
 AppMonEnables, 795
 AppMonLastExeCount, 795
 AppMonLoaded, 795
 CDSData, 796
 CDSState, 796
 CmdCount, 796
 CmdErrCount, 796
 CmdPipe, 797
 CurrentAlivenessState, 797
 CurrentAppMonState, 797
 CurrentCPUHogState, 798
 CurrentCPUHoggingTime, 797
 CurrentCPUUtilIndex, 798
 CurrentEventMonState, 798
 EMTableHandle, 798
 EMTablePtr, 799
 EventMonLoaded, 799
 EventPipe, 799
 EventsMonitoredCount, 799
 ExeCountState, 800
 HkPacket, 800
 MATableHandle, 800
 MATablePtr, 800
 MaxCPUHoggingTime, 801
 MsgActCooldown, 801
 MsgActExec, 801
 MsgActsState, 801
 MyCDSHandle, 802
 RunStatus, 802
 ServiceWatchdogFlag, 802
 SpareBytes, 802
 UtilCpuAvg, 803
 UtilCpuPeak, 803
 UtilizationTracker, 803
 WakeupPipe, 803
HS_AppInit
 hs_app.c, 853
 hs_app.h, 867
HS_AppMain
 hs_app.c, 855
 hs_app.h, 868
HS_AppMonStatusRefresh
 hs_cmds.c, 880
 hs_cmds.h, 901
HS_AppPipe
 hs_cmds.c, 881
 hs_cmds.h, 902
HS_BADEMT_LONG_UNSUB_EID
 CFS Health and Safety Event IDs, 169
HS_BADEMT_SHORT_UNSUB_EID
 CFS Health and Safety Event IDs, 170
HS_BITS_PER_APPMON_ENABLE
 hs_msg.h, 965
HS_CC_ERR_EID
 CFS Health and Safety Event IDs, 170
HS_CDS_CORRUPT_ERR_EID
 CFS Health and Safety Event IDs, 171
HS_CDS_IN_USE
 hs_msgdefs.h, 966
HS_CDS_RESTORE_ERR_EID
 CFS Health and Safety Event IDs, 171
HS_CDSData_t, 804
 MaxResets, 804
 MaxResetsNot, 805

ResetsPerformed, 805
ResetsPerformedNot, 805
HS_CDSNAME
 hs_app.h, 866
HS_CMD_MID
 CFS Health and Safety Command Message IDs, 137
HS_CMD_PIPE_DEPTH
 CFS Health and Safety Platform Configuration, 142
HS_CMD_PIPE_NAME
 hs_app.h, 866
HS_CPU_ALIVE_PERIOD
 CFS Health and Safety Platform Configuration, 142
HS_CPU_ALIVE_STRING
 CFS Health and Safety Platform Configuration, 142
HS_CPUHOG_DEFAULT_STATE
 CFS Health and Safety Platform Configuration, 143
HS_CPUMON_HOGGING_ERR_EID
 CFS Health and Safety Event IDs, 172
HS_CR_CHILD_TASK_ERR_EID
 CFS Health and Safety Event IDs, 172
HS_CR_CMD_PIPE_ERR_EID
 CFS Health and Safety Event IDs, 173
HS_CR_EVENT_PIPE_ERR_EID
 CFS Health and Safety Event IDs, 173
HS_CR_SYNC_CALLBACK_ERR_EID
 CFS Health and Safety Event IDs, 174
HS_CR_WAKEUP_PIPE_ERR_EID
 CFS Health and Safety Event IDs, 174
HS_CUSTOM_INIT_ERR_EID
 CFS Health and Safety Event IDs, 175
HS_CustomCleanup
 hs_custom.c, 919
 hs_custom.h, 932
HS_CustomCommands
 hs_custom.c, 920
 hs_custom.h, 932
HS_CustomData
 hs_custom.c, 929
 hs_custom.h, 941
HS_CustomData_t, 806
 IdleTaskID, 806
 IdleTaskRunStatus, 807
 LastIdleTaskExec, 807
 LastIdleTaskInterval, 807
 ThisIdleTaskExec, 807
 UtilArray, 808
 UtilArrayIndex, 808
 UtilArrayMask, 808
 UtilCallsPerMark, 808
 UtilCycleCounter, 809
 UtilDiv, 809
 UtilMask, 809
 UtilMult1, 809
 UtilMult2, 810
HS_CustomGetUtil
 hs_custom.c, 921
 hs_custom.h, 933
HS_CustomInit
 hs_custom.c, 922
 hs_custom.h, 934
HS_CustomMonitorUtilization
 hs_custom.c, 923
 hs_custom.h, 935
HS_DISABLE_ALIVENESS_CC
 CFS Health and Safety Command Codes, 206
HS_DISABLE_ALIVENESS_DBG_EID
 CFS Health and Safety Event IDs, 175
HS_DISABLE_APPMON_CC
 CFS Health and Safety Command Codes, 207
HS_DISABLE_APPMON_DBG_EID
 CFS Health and Safety Event IDs, 176
HS_DISABLE_APPMON_ERR_EID
 CFS Health and Safety Event IDs, 176
HS_DISABLE_CPUHOG_CC
 CFS Health and Safety Command Codes, 208
HS_DISABLE_CPUHOG_DBG_EID
 CFS Health and Safety Event IDs, 177
HS_DISABLE_EVENTMON_CC
 CFS Health and Safety Command Codes, 209
HS_DISABLE_EVENTMON_DBG_EID
 CFS Health and Safety Event IDs, 177
HS_DISABLE_EVENTMON_ERR_EID
 CFS Health and Safety Event IDs, 178
HS_Default_AppMon_Tbl
 hs_amt.c, 988
HS_Default_EventMon_Tbl
 hs_emt.c, 989
HS_Default_ExeCount_Tbl
 hs_xct.c, 991
HS_Default_MsgActs_Tbl
 hs_mat.c, 990
HS_DisableAlivenessCmd
 hs_cmds.c, 883
 hs_cmds.h, 903
HS_DisableAppMonCmd
 hs_cmds.c, 884
 hs_cmds.h, 904
HS_DisableCPUHogCmd
 hs_cmds.c, 885
 hs_cmds.h, 905
HS_DisableEventMonCmd
 hs_cmds.c, 886
 hs_cmds.h, 906
HS_EM_TBL_NULL_ERR_EID
 CFS Health and Safety Event IDs, 178
HS_EMT_ACT_APP_DELETE
 hs_tbldefs.h, 973
HS_EMT_ACT_APP_RESTART

hs_tbldefs.h, 974
HS_EMT_ACT_LAST_NONMSG
 hs_tbldefs.h, 974
HS_EMT_ACT_MSG
 hs_tbl.h, 969
HS_EMT_ACT_NOACT
 hs_tbldefs.h, 974
HS_EMT_ACT_PROC_RESET
 hs_tbldefs.h, 974
HS_EMT_FILENAME
 CFS Health and Safety Platform Configuration, 143
HS_EMT_LD_ERR_EID
 CFS Health and Safety Event IDs, 179
HS_EMT_REG_ERR_EID
 CFS Health and Safety Event IDs, 179
HS_EMT_TABLENAME
 hs_tbldefs.h, 975
HS_EMTCActionIsValid
 hs_utils.c, 981
 hs_utils.h, 985
HS_EMTEEntry_t, 810
 ActionType, 811
 AppName, 811
 EventID, 811
 NullTerm, 811
HS_EMTVAL_ERR_ACT
 hs_tbldefs.h, 975
HS_EMTVAL_ERR_EID
 CFS Health and Safety Event IDs, 180
HS_EMTVAL_ERR_NUL
 hs_tbldefs.h, 975
HS_EMTVAL_INF_EID
 CFS Health and Safety Event IDs, 180
HS_EMTVAL_NO_ERR
 hs_tbldefs.h, 975
HS_ENABLE_ALIVENESS_CC
 CFS Health and Safety Command Codes, 210
HS_ENABLE_ALIVENESS_DBG_EID
 CFS Health and Safety Event IDs, 181
HS_ENABLE_APPMON_CC
 CFS Health and Safety Command Codes, 211
HS_ENABLE_APPMON_DBG_EID
 CFS Health and Safety Event IDs, 181
HS_ENABLE_CPUHOG_CC
 CFS Health and Safety Command Codes, 212
HS_ENABLE_CPUHOG_DBG_EID
 CFS Health and Safety Event IDs, 182
HS_ENABLE_EVENTMON_CC
 CFS Health and Safety Command Codes, 213
HS_ENABLE_EVENTMON_DBG_EID
 CFS Health and Safety Event IDs, 182
HS_EVENT_PIPE_DEPTH
 CFS Health and Safety Platform Configuration, 143
HS_EVENT_PIPE_NAME
 hs_app.h, 866
HS_EVENTMON_DEFAULT_STATE
 CFS Health and Safety Platform Configuration, 144
HS_EVENTMON_DELETE_ERR_EID
 CFS Health and Safety Event IDs, 183
HS_EVENTMON_GETADDR_ERR_EID
 CFS Health and Safety Event IDs, 183
HS_EVENTMON_LONG_SUB_EID
 CFS Health and Safety Event IDs, 184
HS_EVENTMON_LONG_UNSUB_EID
 CFS Health and Safety Event IDs, 184
HS_EVENTMON_MSGACTS_ERR_EID
 CFS Health and Safety Event IDs, 185
HS_EVENTMON_NOT_DELETED_ERR_EID
 CFS Health and Safety Event IDs, 185
HS_EVENTMON_NOT_RESTARTED_ERR_EID
 CFS Health and Safety Event IDs, 186
HS_EVENTMON_PROC_ERR_EID
 CFS Health and Safety Event IDs, 186
HS_EVENTMON_RESTART_ERR_EID
 CFS Health and Safety Event IDs, 187
HS_EVENTMON_SHORT_SUB_EID
 CFS Health and Safety Event IDs, 187
HS_EVENTMON_SHORT_UNSUB_EID
 CFS Health and Safety Event IDs, 188
HS_EXECOUNT_GETADDR_ERR_EID
 CFS Health and Safety Event IDs, 188
HS_EnableAlivenessCmd
 hs_cmds.c, 887
 hs_cmds.h, 907
HS_EnableAppMonCmd
 hs_cmds.c, 888
 hs_cmds.h, 908
HS_EnableCPUHogCmd
 hs_cmds.c, 889
 hs_cmds.h, 909
HS_EnableEventMonCmd
 hs_cmds.c, 890
 hs_cmds.h, 910
HS_HK_TLM_MID
 CFS Health and Safety Telemetry Message IDs, 138
HS_HKREQ_LEN_ERR_EID
 CFS Health and Safety Event IDs, 189
HS_HKREQ_RESOURCE_DBG_EID
 CFS Health and Safety Event IDs, 189
HS_HkPacket_t, 812
 AppMonEnables, 813
 CmdCount, 813
 CmdErrCount, 813
 CurrentAlivenessState, 813
 CurrentAppMonState, 813
 CurrentCPUHogState, 814
 CurrentEventMonState, 814
 EventsMonitoredCount, 814

InvalidEventMonCount, 814
MaxResets, 815
MsgActExec, 815
ResetsPerformed, 815
SpareBytes, 815
StatusFlags, 816
TlmHeader, 816
UtilCpuAvg, 816
UtilCpuPeak, 816
HS_HousekeepingReq
 hs_cmds.c, 891
 hs_cmds.h, 911
HS_IDLE_TASK_FLAGS
 CFS Health and Safety Platform Configuration, 144
HS_IDLE_TASK_NAME
 CFS Health and Safety Platform Configuration, 144
HS_IDLE_TASK_PRIORITY
 CFS Health and Safety Platform Configuration, 145
HS_IDLE_TASK_STACK_PTR
 CFS Health and Safety Platform Configuration, 145
HS_IDLE_TASK_STACK_SIZE
 CFS Health and Safety Platform Configuration, 145
HS_IDLETASK_PERF_ID
 CFS Health and Safety Mission Configuration, 135
HS_INIT_EID
 CFS Health and Safety Event IDs, 190
HS_INVALID_EXECOUNT
 hs_msgdefs.h, 966
HS_IdleTask
 hs_custom.c, 923
 hs_custom.h, 935
HS_LEN_ERR_EID
 CFS Health and Safety Event IDs, 190
HS_LOADED_AMT
 hs_msgdefs.h, 966
HS_LOADED_EMT
 hs_msgdefs.h, 967
HS_LOADED_MAT
 hs_msgdefs.h, 967
HS_LOADED_XCT
 hs_msgdefs.h, 967
HS_MA_TBL_NULL_ERR_EID
 CFS Health and Safety Event IDs, 191
HS_MAJOR_VERSION
 CFS Health and Safety Version, 222
HS_MAT_FILENAME
 CFS Health and Safety Platform Configuration, 146
HS_MAT_LD_ERR_EID
 CFS Health and Safety Event IDs, 191
HS_MAT_REG_ERR_EID
 CFS Health and Safety Event IDs, 192
HS_MAT_STATE_DISABLED
 hs_tbldefs.h, 976
HS_MAT_STATE_ENABLED
 hs_tbldefs.h, 976
hs_tbldefs.h, 976
HS_MAT_STATE_NOEVENT
 hs_tbldefs.h, 976
HS_MAT_TABLENAME
 hs_tbldefs.h, 976
HS_MATEEntry_t, 817
 Cooldown, 817
 EnableState, 817
 MsgBuf, 818
HS_MATMsgBuf_t, 818
 Buffer, 819
 Message, 819
HS_MATVAL_ERR_EID
 CFS Health and Safety Event IDs, 192
HS_MATVAL_ERR_ENA
 hs_tbldefs.h, 977
HS_MATVAL_ERR_ID
 hs_tbldefs.h, 977
HS_MATVAL_ERR_LEN
 hs_tbldefs.h, 977
HS_MATVAL_INF_EID
 CFS Health and Safety Event IDs, 193
HS_MATVAL_NO_ERR
 hs_tbldefs.h, 977
HS_MAX_EXEC_CNT_SLOTS
 CFS Health and Safety Platform Configuration, 146
HS_MAX_MONITORED_APPS
 CFS Health and Safety Platform Configuration, 146
HS_MAX_MONITORED_EVENTS
 CFS Health and Safety Platform Configuration, 147
HS_MAX_MSG_ACT_SIZE
 CFS Health and Safety Platform Configuration, 147
HS_MAX_MSG_ACT_TYPES
 CFS Health and Safety Platform Configuration, 148
HS_MAX_RESTART_ACTIONS
 CFS Health and Safety Platform Configuration, 148
HS_MID_ERR_EID
 CFS Health and Safety Event IDs, 193
HS_MINOR_VERSION
 CFS Health and Safety Version, 222
HS_MISSION_REV
 CFS Health and Safety Platform Configuration, 149
HS_MSGACTS_GETADDR_ERR_EID
 CFS Health and Safety Event IDs, 194
HS_MarkIdleCallback
 hs_custom.c, 924
 hs_custom.h, 936
HS_MonitorApplications
 hs_monitors.c, 946
 hs_monitors.h, 955
HS_MonitorEvent
 hs_monitors.c, 947
 hs_monitors.h, 956
HS_MonitorUtilization

hs_monitors.c, 948
 hs_monitors.h, 957
HS_MsgActsStatusRefresh
 hs_cmds.c, 893
 hs_cmds.h, 913
HS_NOOP_CC
 CFS Health and Safety Command Codes, 214
HS_NOOP_INF_EID
 CFS Health and Safety Event IDs, 194
HS_NoArgsCmd_t, 819
 CmdHeader, 820
HS_NoopCmd
 hs_cmds.c, 893
 hs_cmds.h, 913
HS_POST_PROCESSING_DELAY
 CFS Health and Safety Platform Configuration, 149
HS_ProcessCommands
 hs_app.c, 857
 hs_app.h, 870
HS_ProcessMain
 hs_app.c, 859
 hs_app.h, 872
HS_REPORT_DIAG_CC
 CFS Health and Safety Command Codes, 215
HS_RESET_CC
 CFS Health and Safety Command Codes, 216
HS_RESET_DBG_EID
 CFS Health and Safety Event IDs, 195
HS_RESET_LIMIT_ERR_EID
 CFS Health and Safety Event IDs, 195
HS_RESET_RESETS_DBG_EID
 CFS Health and Safety Event IDs, 196
HS_RESET_RESETS_PERFORMED_CC
 CFS Health and Safety Command Codes, 217
HS_RESET_TASK_DELAY
 CFS Health and Safety Platform Configuration, 150
HS_REVISION
 CFS Health and Safety Version, 222
HS_ResetCmd
 hs_cmds.c, 894
 hs_cmds.h, 914
HS_ResetCounters
 hs_cmds.c, 895
 hs_cmds.h, 915
HS_ResetResetsPerformedCmd
 hs_cmds.c, 896
 hs_cmds.h, 916
HS_SEND_HK_MID
 CFS Health and Safety Command Message IDs, 137
HS_SET_MAX_RESETS_CC
 CFS Health and Safety Command Codes, 218
HS_SET_MAX_RESETS_DBG_EID
 CFS Health and Safety Event IDs, 196
HS_SET_UTIL_DIAG_CC
 CFS Health and Safety Command Codes, 219
HS_SET_UTIL_DIAG_DBG_EID
 CFS Health and Safety Event IDs, 197
HS_SET_UTIL_PARAMS_CC
 CFS Health and Safety Command Codes, 220
HS_SET_UTIL_PARAMS_DBG_EID
 CFS Health and Safety Event IDs, 197
HS_SET_UTIL_PARAMS_ERR_EID
 CFS Health and Safety Event IDs, 198
HS_STARTUP_SYNC_TIMEOUT
 CFS Health and Safety Platform Configuration, 150
HS_STATE_DISABLED
 hs_msgdefs.h, 967
HS_STATE_ENABLED
 hs_msgdefs.h, 967
HS_SUB_CMD_ERR_EID
 CFS Health and Safety Event IDs, 198
HS_SUB_LONG_EVS_ERR_EID
 CFS Health and Safety Event IDs, 199
HS_SUB_REQ_ERR_EID
 CFS Health and Safety Event IDs, 199
HS_SUB_SHORT_EVS_ERR_EID
 CFS Health and Safety Event IDs, 200
HS_SUB_WAKEUP_ERR_EID
 CFS Health and Safety Event IDs, 200
HS_SblInit
 hs_app.c, 861
 hs_app.h, 874
HS_SetCDSData
 hs_monitors.c, 949
 hs_monitors.h, 958
HS_SetMaxResetsCmd
 hs_cmds.c, 897
 hs_cmds.h, 917
HS_SetMaxResetsCmd_t, 820
 CmdHeader, 820
 MaxResets, 821
 Padding, 821
HS_SetUtilDiagCmd
 hs_custom.c, 925
 hs_custom.h, 937
HS_SetUtilDiagCmd_t, 821
 CmdHeader, 822
 Mask, 822
HS_SetUtilParamsCmd
 hs_custom.c, 926
 hs_custom.h, 938
HS_SetUtilParamsCmd_t, 822
 CmdHeader, 823
 Div, 823
 Mult1, 823
 Mult2, 824
HS_TBL_VAL_ERR
 hs_app.h, 866

HS_TblInit
 hs_app.c, 863
 hs_app.h, 876
HS_UTIL_AVERAGE_NUM_INTERVAL
 CFS Health and Safety Platform Configuration, 151
HS_UTIL_CALLS_PER_MARK
 CFS Health and Safety Platform Configuration, 151
HS_UTIL_CONV_DIV
 CFS Health and Safety Platform Configuration, 152
HS_UTIL_CONV_MULT1
 CFS Health and Safety Platform Configuration, 152
HS_UTIL_CONV_MULT2
 CFS Health and Safety Platform Configuration, 153
HS_UTIL_CYCLES_PER_INTERVAL
 CFS Health and Safety Platform Configuration, 153
HS_UTIL_DIAG_MASK
 CFS Health and Safety Platform Configuration, 154
HS_UTIL_DIAG_REPORT_EID
 CFS Health and Safety Event IDs, 201
HS_UTIL_DIAG_REPORTS
 hs_custom.h, 931
HS_UTIL_HOGGING_TIMEOUT
 CFS Health and Safety Platform Configuration, 154
HS_UTIL_PEAK_NUM_INTERVAL
 CFS Health and Safety Platform Configuration, 154
HS_UTIL_PER_INTERVAL_HOGGING
 CFS Health and Safety Platform Configuration, 155
HS_UTIL_PER_INTERVAL_TOTAL
 CFS Health and Safety Platform Configuration, 155
HS_UTIL_TIME_DIAG_ARRAY_LENGTH
 CFS Health and Safety Platform Configuration, 156
HS_UTIL_TIME_DIAG_ARRAY_MASK
 CFS Health and Safety Platform Configuration, 156
HS_UtilDiagReport
 hs_custom.c, 927
 hs_custom.h, 939
HS_UtilizationIncrement
 hs_custom.c, 928
 hs_custom.h, 940
HS_UtilizationMark
 hs_custom.c, 928
 hs_custom.h, 940
HS_ValidateAMTable
 hs_monitors.c, 950
 hs_monitors.h, 959
HS_ValidateEMTable
 hs_monitors.c, 951
 hs_monitors.h, 960
HS_ValidateMATable
 hs_monitors.c, 952
 hs_monitors.h, 961
HS_ValidateXCTable
 hs_monitors.h, 963
HS_VerifyMsgLength
 hs_utils.c, 982
 hs_utils.h, 985
HS_WAKEUP_MID
 CFS Health and Safety Command Message IDs, 137
HS_WAKEUP_PIPE_DEPTH
 CFS Health and Safety Platform Configuration, 156
HS_WAKEUP_PIPE_NAME
 hs_app.h, 866
HS_WAKEUP_TIMEOUT
 CFS Health and Safety Platform Configuration, 157
HS_WATCHDOG_TIMEOUT_VALUE
 CFS Health and Safety Platform Configuration, 157
HS_XC_TBL_NULL_ERR_EID
 CFS Health and Safety Event IDs, 201
HS_XCT_FILENAME
 CFS Health and Safety Platform Configuration, 158
HS_XCT_LD_ERR_EID
 CFS Health and Safety Event IDs, 202
HS_XCT_REG_ERR_EID
 CFS Health and Safety Event IDs, 202
HS_XCT_TABLENAME
 hs_tbldefs.h, 978
HS_XCT_TYPE_APP_CHILD
 hs_tbldefs.h, 978
HS_XCT_TYPE_APP_MAIN
 hs_tbldefs.h, 978
HS_XCT_TYPE_DEVICE
 hs_tbldefs.h, 978
HS_XCT_TYPE_ISR
 hs_tbldefs.h, 979
HS_XCT_TYPE_NOTYPE
 hs_tbldefs.h, 979
HS_XCTEntry_t, 824
 NullTerm, 825
 ResourceName, 825
 ResourceType, 825
HS_XCTVAL_ERR_EID
 CFS Health and Safety Event IDs, 203
HS_XCTVAL_ERR_NUL
 hs_tbldefs.h, 979
HS_XCTVAL_ERR_TYPE
 hs_tbldefs.h, 979
HS_XCTVAL_INF_EID
 CFS Health and Safety Event IDs, 203
HS_XCTVAL_NO_ERR
 hs_tbldefs.h, 980
Handle
 CFE_ES_CDSRegDumpRec, 597
HeapBlocksFree
 CFE_ES_HousekeepingTlm_Payload, 609
HeapBytesFree
 CFE_ES_HousekeepingTlm_Payload, 609

HeapMaxBlockSize
 CFE_ES_HousekeepingTlm_Payload, 609

HkPacket
 HS_AppData_t, 800

host_module_id
 OS_module_prop_t, 837

hs_amt.c
 __attribute__, 988
 HS_Default_AppMon_Tbl, 988

hs_app.c
 HS_AppData, 864
 HS_AppInit, 853
 HS_AppMain, 855
 HS_ProcessCommands, 857
 HS_ProcessMain, 859
 HS_SbInit, 861
 HS_TblInit, 863

hs_app.h
 HS_AppData, 877
 HS_AppInit, 867
 HS_AppMain, 868
 HS_CDSNAME, 866
 HS_CMD_PIPE_NAME, 866
 HS_EVENT_PIPE_NAME, 866
 HS_ProcessCommands, 870
 HS_ProcessMain, 872
 HS_SbInit, 874
 HS_TBL_VAL_ERR, 866
 HS_TblInit, 876
 HS_WAKEUP_PIPE_NAME, 866

hs_cmds.c
 HS_AcquirePointers, 879
 HS_AppMonStatusRefresh, 880
 HS_AppPipe, 881
 HS_DisableAlivenessCmd, 883
 HS_DisableAppMonCmd, 884
 HS_DisableCPUHogCmd, 885
 HS_DisableEventMonCmd, 886
 HS_EnableAlivenessCmd, 887
 HS_EnableAppMonCmd, 888
 HS_EnableCPUHogCmd, 889
 HS_EnableEventMonCmd, 890
 HS_HousekeepingReq, 891
 HS_MsgActsStatusRefresh, 893
 HS_NoopCmd, 893
 HS_ResetCmd, 894
 HS_ResetCounters, 895
 HS_ResetResetsPerformedCmd, 896
 HS_SetMaxResetsCmd, 897

hs_cmds.h
 HS_AcquirePointers, 899
 HS_AppMonStatusRefresh, 901
 HS_AppPipe, 902
 HS_DisableAlivenessCmd, 903

HS_DisableAppMonCmd, 904
 HS_DisableCPUHogCmd, 905
 HS_DisableEventMonCmd, 906
 HS_EnableAlivenessCmd, 907
 HS_EnableAppMonCmd, 908
 HS_EnableCPUHogCmd, 909
 HS_EnableEventMonCmd, 910
 HS_HousekeepingReq, 911
 HS_MsgActsStatusRefresh, 913
 HS_NoopCmd, 913
 HS_ResetCmd, 914
 HS_ResetCounters, 915
 HS_ResetResetsPerformedCmd, 916
 HS_SetMaxResetsCmd, 917

hs_custom.c
 HS_CustomCleanup, 919
 HS_CustomCommands, 920
 HS_CustomData, 929
 HS_CustomGetUtil, 921
 HS_CustomInit, 922
 HS_CustomMonitorUtilization, 923
 HS_IdleTask, 923
 HS_MarkIdleCallback, 924
 HS_SetUtilDiagCmd, 925
 HS_SetUtilParamsCmd, 926
 HS_UtilDiagReport, 927
 HS_UtilizationIncrement, 928
 HS_UtilizationMark, 928

hs_custom.h
 HS_CustomCleanup, 932
 HS_CustomCommands, 932
 HS_CustomData, 941
 HS_CustomGetUtil, 933
 HS_CustomInit, 934
 HS_CustomMonitorUtilization, 935
 HS_IdleTask, 935
 HS_MarkIdleCallback, 936
 HS_SetUtilDiagCmd, 937
 HS_SetUtilParamsCmd, 938
 HS_UTIL_DIAG_REPORTS, 931
 HS_UtilDiagReport, 939
 HS_UtilizationIncrement, 940
 HS_UtilizationMark, 940

hs_emt.c
 __attribute__, 989
 HS_Default_EventMon_Tbl, 989

hs_mat.c
 __attribute__, 990
 HS_Default_MsgActs_Tbl, 990

hs_monitors.c
 HS_MonitorApplications, 946
 HS_MonitorEvent, 947
 HS_MonitorUtilization, 948
 HS_SetCDSData, 949

HS_ValidateAMTable, 950
HS_ValidateEMTable, 951
HS_ValidateMATable, 952
hs_monitors.h
 HS_MonitorApplications, 955
 HS_MonitorEvent, 956
 HS_MonitorUtilization, 957
 HS_SetCDSData, 958
 HS_ValidateAMTable, 959
 HS_ValidateEMTable, 960
 HS_ValidateMATable, 961
 HS_ValidateXCTable, 963
hs_msg.h
 HS_BITS_PER_APPMON_ENABLE, 965
hs_msgdefs.h
 HS_CDS_IN_USE, 966
 HS_INVALID_EXECOUNT, 966
 HS_LOADED_AMT, 966
 HS_LOADED_EMT, 967
 HS_LOADED_MAT, 967
 HS_LOADED_XCT, 967
 HS_STATE_DISABLED, 967
 HS_STATE_ENABLED, 967
hs_tbl.h
 HS_AMT_ACT_MSG, 969
 HS_EMT_ACT_MSG, 969
hs_tbldefs.h
 HS_AMT_ACT_APP_RESTART, 971
 HS_AMT_ACT_EVENT, 971
 HS_AMT_ACT_LAST_NONMSG, 972
 HS_AMT_ACT_NOACT, 972
 HS_AMT_ACT_PROC_RESET, 972
 HS_AMT_TABLENAME, 972
 HS_AMTVAL_ERR_ACT, 973
 HS_AMTVAL_ERR_NUL, 973
 HS_AMTVAL_NO_ERR, 973
 HS_EMT_ACT_APP_DELETE, 973
 HS_EMT_ACT_APP_RESTART, 974
 HS_EMT_ACT_LAST_NONMSG, 974
 HS_EMT_ACT_NOACT, 974
 HS_EMT_ACT_PROC_RESET, 974
 HS_EMT_TABLENAME, 975
 HS_EMTVAL_ERR_ACT, 975
 HS_EMTVAL_ERR_NUL, 975
 HS_EMTVAL_NO_ERR, 975
 HS_MAT_STATE_DISABLED, 976
 HS_MAT_STATE_ENABLED, 976
 HS_MAT_STATE_NOEVENT, 976
 HS_MAT_TABLENAME, 976
 HS_MATVAL_ERR_ENA, 977
 HS_MATVAL_ERR_ID, 977
 HS_MATVAL_ERR_LEN, 977
 HS_MATVAL_NO_ERR, 977
 HS_XCT_TABLENAME, 978
HS_XCT_TYPE_APP_CHILD, 978
HS_XCT_TYPE_APP_MAIN, 978
HS_XCT_TYPE_DEVICE, 978
HS_XCT_TYPE_ISR, 979
HS_XCT_TYPE_NOTYPE, 979
HS_XCTVAL_ERR_NUL, 979
HS_XCTVAL_ERR_TYPE, 979
HS_XCTVAL_NO_ERR, 980
hs_utils.c
 HS_AMTActionIsValid, 981
 HS_EMTActionIsValid, 981
 HS_VerifyMsgLength, 982
hs_utils.h
 HS_AMTActionIsValid, 984
 HS_EMTActionIsValid, 985
 HS_VerifyMsgLength, 985
hs_xct.c
 __attribute__, 991
 HS_Default_ExeCount_Tbl, 991
IdleTaskID
 HS_CustomData_t, 806
IdleTaskRunStatus
 HS_CustomData_t, 807
InactiveBufferAddr
 CFE_TBL_TblRegPacket_Payload, 752
Index
 CFE_SB_MsgMapFileEntry, 696
int16
 common_types.h, 1446
int32
 common_types.h, 1446
int64
 common_types.h, 1446
int8
 common_types.h, 1446
InternalErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 690
interval_time
 OS_timer_prop_t, 849
intptr
 common_types.h, 1447
InvalidEventMonCount
 HS_HkPacket_t, 814
IsPending
 CFE_FS_FileWriteMetaData, 681
IsValid
 OS_file_prop_t, 829
LENGTHCHECK
 osapi-macros.h, 1473
largest_free_block
 OS_heap_prop_t, 833
LastFileDumped
 CFE_TBL_HousekeepingTlm_Payload, 734

LastFileLoaded
 CFE_TBL_HousekeepingTlm_Payload, 735
 CFE_TBL_Info, 740
 CFE_TBL_TblRegPacket_Payload, 752

LastIdleTaskExec
 HS_CustomData_t, 807

LastIdleTaskInterval
 HS_CustomData_t, 807

LastTableLoaded
 CFE_TBL_HousekeepingTlm_Payload, 735

LastUpdateTime
 CFE_TBL_HousekeepingTlm_Payload, 735

LastUpdatedTable
 CFE_TBL_HousekeepingTlm_Payload, 735

LastValCrc
 CFE_TBL_HousekeepingTlm_Payload, 736

LastValStatus
 CFE_TBL_HousekeepingTlm_Payload, 736

LastValTableName
 CFE_TBL_HousekeepingTlm_Payload, 736

LeapSeconds
 CFE_TIME_HousekeepingTlm_Payload, 773
 CFE_TIME_LeapsCmd_Payload, 776

Length
 CCSDS_PrimaryHeader, 585
 CFE_FS_Header, 683

LoadFilename
 CFE_TBL_LoadCmd_Payload, 743

LoadPending
 CFE_TBL_TblRegPacket_Payload, 753

LocalIntCounter
 CFE_TIME_DiagnosticTlm_Payload, 764

LocalTaskCounter
 CFE_TIME_DiagnosticTlm_Payload, 764

LogEnabled
 CFE_EVS_HousekeepingTlm_Payload, 663

LogFilename
 CFE_EVS_LogFileCmd_Payload, 667

LogFullFlag
 CFE_EVS_HousekeepingTlm_Payload, 663

LogMode
 CFE_EVS_HousekeepingTlm_Payload, 664
 CFE_EVS_SetLogMode_Payload, 675

LogOverflowCounter
 CFE_EVS_HousekeepingTlm_Payload, 664

LongDouble
 CFE_ES_PoolAlign, 625
 CFE_SB_Msg, 694

LongInt
 CFE_ES_PoolAlign, 625
 CFE_SB_Msg, 694

MATableHandle
 HS_AppData_t, 800

MATablePtr
 HS_AppData_t, 800

MainTaskId
 CFE_ES_AppInfo, 590

MainTaskName
 CFE_ES_AppInfo, 590

Mask
 CFE_EVS_AppNameEventIDMaskCmd_Payload,
 655
 CFE_EVS_BinFilter, 658
 HS_SetUtilDiagCmd_t, 822

MaxCPUHoggingTime
 HS_AppData_t, 801

MaxElapsed
 CFE_TIME_DiagnosticTlm_Payload, 764

MaxFds
 os_fsinfo_t, 831

MaxLocalClock
 CFE_TIME_DiagnosticTlm_Payload, 765

MaxMemAllowed
 CFE_SB_StatsTlm_Payload, 711

MaxMsgIdsAllowed
 CFE_SB_StatsTlm_Payload, 711

MaxPRCount
 CFE_ES_SetMaxPRCountCmd_Payload, 633

MaxPipeDepthAllowed
 CFE_SB_StatsTlm_Payload, 711

MaxPipesAllowed
 CFE_SB_StatsTlm_Payload, 712

MaxProcessorResets
 CFE_ES_HousekeepingTlm_Payload, 610

MaxQueueDepth
 CFE_SB_PipeDepthStats, 697
 CFE_SB_PipeInfoEntry, 700

MaxResets
 HS_CDSData_t, 804
 HS_HkPacket_t, 815
 HS_SetMaxResetsCmd_t, 821

MaxResetsNot
 HS_CDSData_t, 805

MaxSubscriptionsAllowed
 CFE_SB_StatsTlm_Payload, 712

MaxVolumes
 os_fsinfo_t, 831

MemInUse
 CFE_SB_HousekeepingTlm_Payload, 690
 CFE_SB_StatsTlm_Payload, 712

MemPoolHandle
 CFE_SB_HousekeepingTlm_Payload, 690
 CFE_TBL_HousekeepingTlm_Payload, 736

Message
 CFE_EVS_LongEventTlm_Payload, 669
 HS_MATMMsgBuf_t, 819

MessageFormatMode

CFE_EVS_HousekeepingTlm_Payload, 664
MessageSendCounter
 CFE_EVS_HousekeepingTlm_Payload, 664
MessageTruncCounter
 CFE_EVS_HousekeepingTlm_Payload, 665
MicroSeconds
 CFE_TIME_TimeCmd_Payload, 787
MinElapsed
 CFE_TIME_DiagnosticTlm_Payload, 765
Mode
 CFE_ES_OverWriteSysLogCmd_Payload, 624
Module
 OS_static_symbol_record_t, 843
Msg
 CFE_SB_Msg, 694
MsgActCooldown
 HS_AppData_t, 801
MsgActExec
 HS_AppData_t, 801
 HS_HkPacket_t, 815
MsgActsState
 HS_AppData_t, 801
MsgBuf
 HS_MATEEntry_t, 818
MsgCnt
 CFE_SB_RoutingFileEntry, 705
MsgFormat
 CFE_EVS_SetEventFormatCode_Payload, 673
MsgId
 CFE_SB_MsgMapFileEntry, 696
 CFE_SB_RouteCmd_Payload, 704
 CFE_SB_RoutingFileEntry, 706
 CFE_SB_SingleSubscriptionTlm_Payload, 708
 CFE_SB_SubEntries, 716
MsgIdsInUse
 CFE_SB_StatsTlm_Payload, 712
MsgLimitErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 691
MsgReceiveErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 691
MsgSendErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 691
Mult1
 HS_SetUtilParamsCmd_t, 823
Mult2
 HS_SetUtilParamsCmd_t, 824
MyCDSHandle
 HS_AppData_t, 802

Name
 CFE_ES_AppInfo, 590
 CFE_ES_CDSRegDumpRec, 597
 CFE_TBL_TblRegPacket_Payload, 753
 OS_static_symbol_record_t, 843

name
 OS_bin_sem_prop_t, 826
 OS_count_sem_prop_t, 827
 OS_module_prop_t, 837
 OS_mut_sem_prop_t, 838
 OS_queue_prop_t, 839
 OS_socket_prop_t, 842
 OS_task_prop_t, 845
 OS_timebase_prop_t, 848
 OS_timer_prop_t, 849
NoSubscribersCounter
 CFE_SB_HousekeepingTlm_Payload, 691
nominal_interval_time
 OS_timebase_prop_t, 848
NullTerm
 HS_AMTEEntry_t, 792
 HS_EMTEEntry_t, 811
 HS_XCTEEntry_t, 825
NumBlocksRequested
 CFE_ES_MemPoolStats, 618
NumBytes
 CFE_TBL_File_Hdr, 729
NumCreated
 CFE_ES_BlockStats, 596
NumFree
 CFE_ES_BlockStats, 596
NumFreeBytes
 CFE_ES_MemPoolStats, 619
NumFreeSharedBufs
 CFE_TBL_HousekeepingTlm_Payload, 737
NumLoadPending
 CFE_TBL_HousekeepingTlm_Payload, 737
NumOfChildTasks
 CFE_ES_AppInfo, 591
NumTables
 CFE_TBL_HousekeepingTlm_Payload, 737
NumUsers
 CFE_TBL_Info, 741
NumValRequests
 CFE_TBL_HousekeepingTlm_Payload, 737

OS_API_Init
 OSAL Core Operation APIs, 463
OS_API_Teardown
 OSAL Core Operation APIs, 464
OS_Application_Run
 OSAL Core Operation APIs, 464
OS_Application_Startup
 OSAL Core Operation APIs, 464
OS_ApplicationExit
 OSAL Core Operation APIs, 464
OS_ApplicationShutdown
 OSAL Core Operation APIs, 465
OS_ArgCallback_t

common_types.h, [1447](#)
OS_BSP_GetArgC
 OSAL BSP low level access APIs, [450](#)
OS_BSP_GetArgV
 OSAL BSP low level access APIs, [450](#)
OS_BSP_GetResourceTypeConfig
 OSAL BSP low level access APIs, [450](#)
OS_BSP_SetExitCode
 OSAL BSP low level access APIs, [450](#)
OS_BSP_SetResourceTypeConfig
 OSAL BSP low level access APIs, [450](#)
OS_BUFFER_MSG_DEPTH
 osconfig-example.h, [993](#)
OS_BUFFER_SIZE
 osconfig-example.h, [993](#)
OS_BUILD_BASELINE
 osapi-version.h, [1488](#)
OS_BUILD_NUMBER
 osapi-version.h, [1489](#)
OS_BinSemCreate
 OSAL Binary Semaphore APIs, [444](#)
OS_BinSemDelete
 OSAL Binary Semaphore APIs, [445](#)
OS_BinSemFlush
 OSAL Binary Semaphore APIs, [445](#)
OS_BinSemGetIdByName
 OSAL Binary Semaphore APIs, [446](#)
OS_BinSemGetInfo
 OSAL Binary Semaphore APIs, [447](#)
OS_BinSemGive
 OSAL Binary Semaphore APIs, [447](#)
OS_BinSemTake
 OSAL Binary Semaphore APIs, [448](#)
OS_BinSemTimedWait
 OSAL Binary Semaphore APIs, [448](#)
OS_CHECK
 osapi-constants.h, [1457](#)
OS_CHK_ONLY
 osapi-fs.h, [1468](#)
OS_CloseAllFiles
 OSAL Standard File APIs, [495](#)
OS_CloseFileByName
 OSAL Standard File APIs, [495](#)
OS_ConvertToArrayIndex
 OSAL Object ID Utility APIs, [520](#)
OS_CountSemCreate
 OSAL Counting Semaphore APIs, [467](#)
OS_CountSemDelete
 OSAL Counting Semaphore APIs, [468](#)
OS_CountSemGetIdByName
 OSAL Counting Semaphore APIs, [469](#)
OS_CountSemGetInfo
 OSAL Counting Semaphore APIs, [469](#)
OS_CountSemGive

OSAL Counting Semaphore APIs, [470](#)
OS_CountSemTake
 OSAL Counting Semaphore APIs, [470](#)
OS_CountSemTimedWait
 OSAL Counting Semaphore APIs, [472](#)
OS_DIRENTRY_NAME
 osapi-dir.h, [1459](#)
OS_DeleteAllObjects
 OSAL Core Operation APIs, [465](#)
OS_DirectoryClose
 OSAL Directory APIs, [473](#)
OS_DirectoryOpen
 OSAL Directory APIs, [474](#)
OS_DirectoryRead
 OSAL Directory APIs, [474](#)
OS_DirectoryRewind
 OSAL Directory APIs, [475](#)
OS_ERR_BAD_ADDRESS
 OSAL Return Code Defines, [480](#)
OS_ERR_FILE
 OSAL Return Code Defines, [480](#)
OS_ERR_INCORRECT_OBJ_STATE
 OSAL Return Code Defines, [480](#)
OS_ERR_INCORRECT_OBJ_TYPE
 OSAL Return Code Defines, [481](#)
OS_ERR_INVALID_ARGUMENT
 OSAL Return Code Defines, [481](#)
OS_ERR_INVALID_ID
 OSAL Return Code Defines, [481](#)
OS_ERR_INVALID_PRIORITY
 OSAL Return Code Defines, [481](#)
OS_ERR_INVALID_SIZE
 OSAL Return Code Defines, [481](#)
OS_ERR_NAME_NOT_FOUND
 OSAL Return Code Defines, [482](#)
OS_ERR_NAME_TAKEN
 OSAL Return Code Defines, [482](#)
OS_ERR_NAME_TOO_LONG
 OSAL Return Code Defines, [482](#)
OS_ERR_NO_FREE_IDS
 OSAL Return Code Defines, [482](#)
OS_ERR_NOT_IMPLEMENTED
 OSAL Return Code Defines, [482](#)
OS_ERR_OBJECT_IN_USE
 OSAL Return Code Defines, [483](#)
OS_ERR_OPERATION_NOT_SUPPORTED
 OSAL Return Code Defines, [483](#)
OS_ERR_OUTPUT_TOO_LARGE
 OSAL Return Code Defines, [483](#)
OS_ERR_SEM_NOT_FULL
 OSAL Return Code Defines, [483](#)
OS_ERR_STREAM_DISCONNECTED
 OSAL Return Code Defines, [483](#)
OS_ERROR_ADDRESS_MISALIGNED

OSAL Return Code Defines, 484
OS_ERROR_NAME_LENGTH
 osapi-error.h, 1462
OS_ERROR_TIMEOUT
 OSAL Return Code Defines, 484
OS_ERROR
 OSAL Return Code Defines, 484
OS_Event_t
 osapi-common.h, 1456
OS_EventHandler_t
 osapi-common.h, 1455
OS_FDGetInfo
 OSAL Standard File APIs, 496
OS_FILESTAT_EXEC
 osapi-file.h, 1465
OS_FILESTAT_ISDIR
 osapi-file.h, 1465
OS_FILESTAT_MODE
 osapi-file.h, 1465
OS_FILESTAT_READ
 osapi-file.h, 1465
OS_FILESTAT_SIZE
 osapi-file.h, 1465
OS_FILESTAT_TIME
 osapi-file.h, 1466
OS_FILESTAT_WRITE
 osapi-file.h, 1466
OS_FP_ENABLED
 osapi-task.h, 1483
OS_FS_DEV_NAME_LEN
 osconfig-example.h, 993
OS_FS_ERR_DEVICE_NOT_FREE
 OSAL Return Code Defines, 484
OS_FS_ERR_DRIVE_NOT_CREATED
 OSAL Return Code Defines, 484
OS_FS_ERR_NAME_TOO_LONG
 OSAL Return Code Defines, 485
OS_FS_ERR_PATH_INVALID
 OSAL Return Code Defines, 485
OS_FS_ERR_PATH_TOO_LONG
 OSAL Return Code Defines, 485
OS_FS_GetPhysDriveName
 OSAL File System Level APIs, 508
OS_FS_PHYS_NAME_LEN
 osconfig-example.h, 993
OS_FS_VOL_NAME_LEN
 osconfig-example.h, 994
OS_FdSet, 828
 object_ids, 829
OS_FileOpenCheck
 OSAL Standard File APIs, 497
OS_FileSysAddFixedMap
 OSAL File System Level APIs, 507
OS_FileSysStatVolume
 OSAL File System Level APIs, 508
OS.ForEachObject
 OSAL Object ID Utility APIs, 521
OS.ForEachObjectType
 OSAL Object ID Utility APIs, 521
OS.GetBuildNumber
 osapi-version.h, 1491
OS.GetErrorName
 OSAL Error Info APIs, 489
OS.GetFsInfo
 OSAL File System Level APIs, 509
OS.GetLocalTime
 OSAL Real Time Clock APIs, 451
OS.GetResourceName
 OSAL Object ID Utility APIs, 522
OS.GetVersionCodeName
 osapi-version.h, 1491
OS.GetVersionNumber
 osapi-version.h, 1492
OS.GetVersionString
 osapi-version.h, 1492
OS.HeapGetInfo
 OSAL Heap APIs, 515
OS.INVALID_INT_NUM
 OSAL Return Code Defines, 485
OS.INVALID_POINTER
 OSAL Return Code Defines, 485
OS.INVALID_SEM_VALUE
 OSAL Return Code Defines, 486
OS.IdentifyObject
 OSAL Object ID Utility APIs, 523
OS.IdleLoop
 OSAL Core Operation APIs, 465
OS.MAJOR_VERSION
 osapi-version.h, 1489
OS.MAX_API_NAME
 osconfig-example.h, 994
OS.MAX_BIN_SEMAPHORES
 osconfig-example.h, 994
OS.MAX_CMD_LEN
 osconfig-example.h, 994
OS.MAX_CONSOLES
 osconfig-example.h, 995
OS.MAX_COUNT_SEMAPHORES
 osconfig-example.h, 995
OS.MAX_FILE_NAME
 osconfig-example.h, 995
OS.MAX_FILE_SYSTEMS
 osconfig-example.h, 995
OS.MAX_LOCAL_PATH_LEN
 osapi-constants.h, 1457
OS.MAX_MODULES
 osconfig-example.h, 996
OS.MAX_MUTEXES

osconfig-example.h, 996
OS_MAX_NUM_OPEN_DIRS
 osconfig-example.h, 996
OS_MAX_NUM_OPEN_FILES
 osconfig-example.h, 996
OS_MAX_PATH_LEN
 osconfig-example.h, 997
OS_MAX_QUEUES
 osconfig-example.h, 997
OS_MAX_SYM_LEN
 osconfig-example.h, 997
OS_MAX_TASK_PRIORITY
 osapi-task.h, 1483
OS_MAX_TASKS
 osconfig-example.h, 997
OS_MAX_TIMEBASES
 osconfig-example.h, 998
OS_MAX_TIMERS
 osconfig-example.h, 998
OS_MINOR_VERSION
 osapi-version.h, 1489
OS_MISSION_REV
 osapi-version.h, 1489
OS_MODULE_FILE_EXTENSION
 osconfig-example.h, 998
OS_MODULE_FLAG_GLOBAL_SYMBOLS
 osapi-module.h, 1474
OS_MODULE_FLAG_LOCAL_SYMBOLS
 osapi-module.h, 1474
OS_ModuleInfo
 OSAL Dynamic Loader and Symbol APIs, 527
OS_ModuleLoad
 OSAL Dynamic Loader and Symbol APIs, 528
OS_ModuleSymbolLookup
 OSAL Dynamic Loader and Symbol APIs, 528
OS_ModuleUnload
 OSAL Dynamic Loader and Symbol APIs, 529
OS_MutSemCreate
 OSAL Mutex APIs, 532
OS_MutSemDelete
 OSAL Mutex APIs, 533
OS_MutSemGetIdByName
 OSAL Mutex APIs, 533
OS_MutSemGetInfo
 OSAL Mutex APIs, 534
OS_MutSemGive
 OSAL Mutex APIs, 534
OS_MutSemTake
 OSAL Mutex APIs, 535
OS_NetworkGetHostName
 OSAL Network ID APIs, 537
OS_NetworkGetID
 OSAL Network ID APIs, 537
OS_OBJECT_CREATOR_ANY

osapi-constants.h, 1457
OS_OBJECT_ID_UNDEFINED
 osapi-constants.h, 1458
OS_OBJECT_INDEX_MASK
 osapi-idmap.h, 1471
OS_OBJECT_TYPE_OS_BINSEM
 OSAL Object Type Defines, 516
OS_OBJECT_TYPE_OS_CONSOLE
 OSAL Object Type Defines, 516
OS_OBJECT_TYPE_OS_COUNTSEM
 OSAL Object Type Defines, 517
OS_OBJECT_TYPE_OS_DIR
 OSAL Object Type Defines, 517
OS_OBJECT_TYPE_OS_FILESYS
 OSAL Object Type Defines, 517
OS_OBJECT_TYPE_OS_MODULE
 OSAL Object Type Defines, 517
OS_OBJECT_TYPE_OS_MUTEX
 OSAL Object Type Defines, 517
OS_OBJECT_TYPE_OS_QUEUE
 OSAL Object Type Defines, 518
OS_OBJECT_TYPE_OS_STREAM
 OSAL Object Type Defines, 518
OS_OBJECT_TYPE_OS_TASK
 OSAL Object Type Defines, 518
OS_OBJECT_TYPE_OS_TIMEBASE
 OSAL Object Type Defines, 518
OS_OBJECT_TYPE_OS_TIMECB
 OSAL Object Type Defines, 518
OS_OBJECT_TYPE_SHIFT
 osapi-idmap.h, 1471
OS_OBJECT_TYPE_UNDEFINED
 OSAL Object Type Defines, 519
OS_OBJECT_TYPE_USER
 OSAL Object Type Defines, 519
OS_ObjectIdDefined
 OSAL Object ID Utility APIs, 523
OS_ObjectIdEqual
 OSAL Object ID Utility APIs, 524
OS_ObjectIdFromInteger
 OSAL Object ID Utility APIs, 524
OS_ObjectIdToArrayIndex
 OSAL Object ID Utility APIs, 525
OS_ObjectIdToInteger
 OSAL Object ID Utility APIs, 525
OS_OpenCreate
 OSAL Standard File APIs, 499
OS_PEND
 osapi-constants.h, 1458
OS_PRINTF_CONSOLE_NAME
 osconfig-example.h, 999
OS_PRINTF
 cfe_es.h, 1111
 common_types.h, 1444

OS_QUEUE_EMPTY
 OSAL Return Code Defines, [486](#)

OS_QUEUE_FULL
 OSAL Return Code Defines, [486](#)

OS_QUEUE_ID_ERROR
 OSAL Return Code Defines, [486](#)

OS_QUEUE_INVALID_SIZE
 OSAL Return Code Defines, [486](#)

OS_QUEUE_MAX_DEPTH
 osconfig-example.h, [999](#)

OS_QUEUE_TIMEOUT
 OSAL Return Code Defines, [487](#)

OS_QueueCreate
 OSAL Message Queue APIs, [541](#)

OS_QueueDelete
 OSAL Message Queue APIs, [542](#)

OS_QueueGet
 OSAL Message Queue APIs, [543](#)

OS_QueueGetIdByName
 OSAL Message Queue APIs, [543](#)

OS_QueueGetInfo
 OSAL Message Queue APIs, [544](#)

OS_QueuePut
 OSAL Message Queue APIs, [544](#)

OS_READ_ONLY
 OSAL File Access Option Defines, [491](#)

OS_READ_WRITE
 OSAL File Access Option Defines, [491](#)

OS_REPAIR
 osapi-fs.h, [1468](#)

OS_REVISION
 osapi-version.h, [1489](#)

OS_RegisterEventHandler
 OSAL Core Operation APIs, [466](#)

OS_SEEK_CUR
 OSAL Reference Point For Seek Offset Defines, [492](#)

OS_SEEK_END
 OSAL Reference Point For Seek Offset Defines, [492](#)

OS_SEEK_SET
 OSAL Reference Point For Seek Offset Defines, [492](#)

OS_SEM_EMPTY
 OSAL Semaphore State Defines, [443](#)

OS_SEM_FAILURE
 OSAL Return Code Defines, [487](#)

OS_SEM_FULL
 OSAL Semaphore State Defines, [443](#)

OS_SEM_TIMEOUT
 OSAL Return Code Defines, [487](#)

OS_SHELL_CMD_INPUT_FILE_NAME
 osconfig-example.h, [999](#)

OS SOCKADDR_MAX_LEN
 osapi-sockets.h, [1481](#)
 osconfig-example.h, [999](#)

OS_STR_HELPER
 osapi-version.h, [1490](#)

OS_STR
 osapi-version.h, [1490](#)

OS_SUCCESS
 OSAL Return Code Defines, [487](#)

OS_SelectFdAdd
 OSAL Select APIs, [546](#)

OS_SelectFdClear
 OSAL Select APIs, [547](#)

OS_SelectFdIsSet
 OSAL Select APIs, [547](#)

OS_SelectFdZero
 OSAL Select APIs, [548](#)

OS_SelectMultiple
 OSAL Select APIs, [548](#)

OS_SelectSingle
 OSAL Select APIs, [549](#)

OS_SetLocalTime
 OSAL Real Time Clock APIs, [452](#)

OS_ShellOutputToFile
 OSAL Shell APIs, [551](#)

OS_SockAddr_t, [839](#)
 ActualLength, [840](#)
 AddrData, [840](#)
 OS_SockAddrData_t, [840](#)
 AlignPtr, [841](#)
 AlignU32, [841](#)
 Buffer, [841](#)

OS_SocketAccept
 OSAL Socket Management APIs, [556](#)

OS_SocketAddrFromString
 OSAL Socket Address APIs, [552](#)

OS_SocketAddrGetPort
 OSAL Socket Address APIs, [553](#)

OS_SocketAddrInit
 OSAL Socket Address APIs, [553](#)

OS_SocketAddrSetPort
 OSAL Socket Address APIs, [554](#)

OS_SocketAddrToString
 OSAL Socket Address APIs, [555](#)

OS_SocketBind
 OSAL Socket Management APIs, [558](#)

OS_SocketConnect
 OSAL Socket Management APIs, [559](#)

OS_SocketDomain_t
 osapi-sockets.h, [1481](#)

OS_SocketGetIdByName
 OSAL Socket Management APIs, [559](#)

OS_SocketGetInfo
 OSAL Socket Management APIs, [560](#)

OS_SocketOpen
 OSAL Socket Management APIs, [561](#)

OS_SocketRecvFrom
 OSAL Socket Management APIs, [561](#)

OS_SocketSendTo
 OSAL Socket Management APIs, 562

OS_SocketShutdown
 OSAL Socket Management APIs, 563

OS_SocketShutdownMode_t
 osapi-sockets.h, 1481

OS_SocketType_t
 osapi-sockets.h, 1482

OS_StatusToInteger
 OSAL Error Info APIs, 489

OS_StreamState_t
 osapi-select.h, 1478

OS_SymbolLookup
 OSAL Dynamic Loader and Symbol APIs, 530

OS_SymbolTableDump
 OSAL Dynamic Loader and Symbol APIs, 530

OS_TIMER_ERR_INTERNAL
 OSAL Return Code Defines, 487

OS_TIMER_ERR_INVALID_ARGS
 OSAL Return Code Defines, 488

OS_TIMER_ERR_TIMER_ID
 OSAL Return Code Defines, 488

OS_TIMER_ERR_UNAVAILABLE
 OSAL Return Code Defines, 488

OS_TaskCreate
 OSAL Task APIs, 564

OS_TaskDelay
 OSAL Task APIs, 565

OS_TaskDelete
 OSAL Task APIs, 566

OS_TaskExit
 OSAL Task APIs, 566

OS_TaskFindIdBySystemData
 OSAL Task APIs, 566

OS_TaskGetId
 OSAL Task APIs, 567

OS_TaskGetIdByName
 OSAL Task APIs, 567

OS_TaskGetInfo
 OSAL Task APIs, 568

OS_TaskInstallDeleteHandler
 OSAL Task APIs, 569

OS_TaskSetPriority
 OSAL Task APIs, 569

OS_TimeAdd
 OSAL Real Time Clock APIs, 453

OS_TimeAssembleFromMicroseconds
 OSAL Real Time Clock APIs, 453

OS_TimeAssembleFromMilliseconds
 OSAL Real Time Clock APIs, 454

OS_TimeAssembleFromNanoseconds
 OSAL Real Time Clock APIs, 454

OS_TimeAssembleFromSubseconds
 OSAL Real Time Clock APIs, 455

OS_TimeBaseCreate
 OSAL Time Base APIs, 571

OS_TimeBaseDelete
 OSAL Time Base APIs, 572

OS_TimeBaseGetFreeRun
 OSAL Time Base APIs, 573

OS_TimeBaseGetIdByName
 OSAL Time Base APIs, 574

OS_TimeBaseGetInfo
 OSAL Time Base APIs, 574

OS_TimeBaseSet
 OSAL Time Base APIs, 575

OS_TimeGetFractionalPart
 OSAL Real Time Clock APIs, 456

OS_TimeGetMicrosecondsPart
 OSAL Real Time Clock APIs, 456

OS_TimeGetMillisecondsPart
 OSAL Real Time Clock APIs, 457

OS_TimeGetNanosecondsPart
 OSAL Real Time Clock APIs, 458

OS_TimeGetSubsecondsPart
 OSAL Real Time Clock APIs, 459

OS_TimeGetTotalMicroseconds
 OSAL Real Time Clock APIs, 459

OS_TimeGetTotalMilliseconds
 OSAL Real Time Clock APIs, 460

OS_TimeGetTotalNanoseconds
 OSAL Real Time Clock APIs, 460

OS_TimeGetTotalSeconds
 OSAL Real Time Clock APIs, 461

OS_TimeSubtract
 OSAL Real Time Clock APIs, 462

OS_TimedRead
 OSAL Standard File APIs, 502

OS_TimedWrite
 OSAL Standard File APIs, 503

OS_TimerAdd
 OSAL Timer APIs, 577

OS_TimerCallback_t
 osapi-timer.h, 1487

OS_TimerCreate
 OSAL Timer APIs, 579

OS_TimerDelete
 OSAL Timer APIs, 580

OS_TimerGetIdByName
 OSAL Timer APIs, 581

OS_TimerGetInfo
 OSAL Timer APIs, 582

OS_TimerSet
 OSAL Timer APIs, 583

OS_TimerSync_t
 osapi-timebase.h, 1486

OS_TranslatePath
 OSAL File System Level APIs, 513

OS_USED
 common_types.h, 1444

OS_UTILITYTASK_PRIORITY
 osconfig-example.h, 1000

OS_UTILITYTASK_STACK_SIZE
 osconfig-example.h, 1000

OS_VERSION_CODENAME
 osapi-version.h, 1490

OS_VERSION_STRING
 osapi-version.h, 1490

OS_VERSION
 osapi-version.h, 1490

OS_WRITE_ONLY
 OSAL File Access Option Defines, 491

OS_bin_sem_prop_t, 825
 creator, 826
 name, 826
 value, 826

OS_chkfs
 OSAL File System Level APIs, 506

OS_chmod
 OSAL Standard File APIs, 493

OS_close
 OSAL Standard File APIs, 494

OS_count_sem_prop_t, 826
 creator, 827
 name, 827
 value, 827

OS_cp
 OSAL Standard File APIs, 496

OS_file_flag_t
 osapi-file.h, 1467

OS_file_prop_t, 829
 IsValid, 829
 Path, 829
 User, 830

OS_heap_prop_t, 833
 free_blocks, 833
 free_bytes, 833
 largest_free_block, 833

OS_initfs
 OSAL File System Level APIs, 510

OS_lseek
 OSAL Standard File APIs, 497

OS_mkdir
 OSAL Directory APIs, 475

OS_mkfs
 OSAL File System Level APIs, 511

OS_module_address_t, 834
 bss_address, 834
 bss_size, 834
 code_address, 835
 code_size, 835
 data_address, 835

 data_size, 835
 flags, 835
 valid, 835

OS_module_prop_t, 836
 addr, 836
 entry_point, 836
 filename, 837
 host_module_id, 837
 name, 837

OS_mount
 OSAL File System Level APIs, 511

OS_mut_sem_prop_t, 837
 creator, 838
 name, 838

OS_mv
 OSAL Standard File APIs, 498

OS_printf
 OSAL Printf APIs, 539

OS_printf_disable
 OSAL Printf APIs, 539

OS_printf_enable
 OSAL Printf APIs, 540

OS_queue_prop_t, 838
 creator, 839
 name, 839

OS_read
 OSAL Standard File APIs, 500

OS_remove
 OSAL Standard File APIs, 500

OS_rename
 OSAL Standard File APIs, 501

OS_rmdir
 OSAL Directory APIs, 476

OS_rmfs
 OSAL File System Level APIs, 512

OS_socket_prop_t, 841
 creator, 842
 name, 842

OS_stat
 OSAL Standard File APIs, 502

OS_static_symbol_record_t, 842
 Address, 843
 Module, 843
 Name, 843

OS_statvfs_t, 843
 block_size, 844
 blocks_free, 844
 total_blocks, 844

OS_task_prop_t, 845
 creator, 845
 name, 845
 priority, 845
 stack_size, 845

OS_time_t, 846

ticks, [846](#)
OS_timebase_prop_t, [847](#)
 accuracy, [847](#)
 creator, [847](#)
 freerun_time, [848](#)
 name, [848](#)
 nominal_interval_time, [848](#)
OS_timer_prop_t, [848](#)
 accuracy, [849](#)
 creator, [849](#)
 interval_time, [849](#)
 name, [849](#)
 start_time, [849](#)
OS_unmount
 OSAL File System Level APIs, [513](#)
OS_write
 OSAL Standard File APIs, [504](#)
OSAL BSP low level access APIs, [450](#)
 OS_BSP_GetArgC, [450](#)
 OS_BSP_GetArgV, [450](#)
 OS_BSP_GetResourceTypeConfig, [450](#)
 OS_BSP_SetExitCode, [450](#)
 OS_BSP_SetResourceTypeConfig, [450](#)
OSAL Binary Semaphore APIs, [444](#)
 OS_BinSemCreate, [444](#)
 OS_BinSemDelete, [445](#)
 OS_BinSemFlush, [445](#)
 OS_BinSemGetIdByName, [446](#)
 OS_BinSemGetInfo, [447](#)
 OS_BinSemGive, [447](#)
 OS_BinSemTake, [448](#)
 OS_BinSemTimedWait, [448](#)
OSAL Core Operation APIs, [463](#)
 OS_API_Init, [463](#)
 OS_API_Teardown, [464](#)
 OS_Application_Run, [464](#)
 OS_Application_Startup, [464](#)
 OS_ApplicationExit, [464](#)
 OS_ApplicationShutdown, [465](#)
 OS_DeleteAllObjects, [465](#)
 OS_IdleLoop, [465](#)
 OS_RegisterEventHandler, [466](#)
OSAL Counting Semaphore APIs, [467](#)
 OS_CountSemCreate, [467](#)
 OS_CountSemDelete, [468](#)
 OS_CountSemGetIdByName, [469](#)
 OS_CountSemGetInfo, [469](#)
 OS_CountSemGive, [470](#)
 OS_CountSemTake, [470](#)
 OS_CountSemTimedWait, [472](#)
OSAL Directory APIs, [473](#)
 OS_DirectoryClose, [473](#)
 OS_DirectoryOpen, [474](#)
 OS_DirectoryRead, [474](#)
 OS_DirectoryRewind, [475](#)
 OS_mkdir, [475](#)
 OS_rmdir, [476](#)
OSAL Dynamic Loader and Symbol APIs, [527](#)
 OS_ModuleInfo, [527](#)
 OS_ModuleLoad, [528](#)
 OS_ModuleSymbolLookup, [528](#)
 OS_ModuleUnload, [529](#)
 OS_SymbolLookup, [530](#)
 OS_SymbolTableDump, [530](#)
OSAL Error Info APIs, [489](#)
 OS_GetErrorName, [489](#)
 OS_StatusToInteger, [489](#)
OSAL File Access Option Defines, [491](#)
 OS_READ_ONLY, [491](#)
 OS_READ_WRITE, [491](#)
 OS_WRITE_ONLY, [491](#)
OSAL File System Level APIs, [506](#)
 OS_FS_GetPhysDriveName, [508](#)
 OS_FileSysAddFixedMap, [507](#)
 OS_FileSysStatVolume, [508](#)
 OS_GetFslInfo, [509](#)
 OS_TranslatePath, [513](#)
 OS_chkfs, [506](#)
 OS_initfs, [510](#)
 OS_mkfs, [511](#)
 OS_mount, [511](#)
 OS_rmfs, [512](#)
 OSUnmount, [513](#)
OSAL Heap APIs, [515](#)
 OS_HeapGetInfo, [515](#)
OSAL Message Queue APIs, [541](#)
 OS_QueueCreate, [541](#)
 OS_QueueDelete, [542](#)
 OS_QueueGet, [543](#)
 OS_QueueGetIdByName, [543](#)
 OS_QueueGetInfo, [544](#)
 OS_QueuePut, [544](#)
OSAL Mutex APIs, [532](#)
 OS_MutSemCreate, [532](#)
 OS_MutSemDelete, [533](#)
 OS_MutSemGetIdByName, [533](#)
 OS_MutSemGetInfo, [534](#)
 OS_MutSemGive, [534](#)
 OS_MutSemTake, [535](#)
OSAL Network ID APIs, [537](#)
 OS_NetworkGetHostName, [537](#)
 OS_NetworkGetID, [537](#)
OSAL Object ID Utility APIs, [520](#)
 OS_ConvertToArrayIndex, [520](#)
 OS_ForEachObject, [521](#)
 OS_ForEachObjectType, [521](#)
 OS_GetResourceName, [522](#)
 OS_IdentifyObject, [523](#)

OS_ObjectIdDefined, 523
OS_ObjectIdEqual, 524
OS_ObjectIdFromInteger, 524
OS_ObjectIdToArrayIndex, 525
OS_ObjectIdToInteger, 525
OSAL Object Type Defines, 516
 OS_OBJECT_TYPE_OS_BINSEM, 516
 OS_OBJECT_TYPE_OS_CONSOLE, 516
 OS_OBJECT_TYPE_OS_COUNTSEM, 517
 OS_OBJECT_TYPE_OS_DIR, 517
 OS_OBJECT_TYPE_OS_FILESYS, 517
 OS_OBJECT_TYPE_OS_MODULE, 517
 OS_OBJECT_TYPE_OS_MUTEX, 517
 OS_OBJECT_TYPE_OS_QUEUE, 518
 OS_OBJECT_TYPE_OS_STREAM, 518
 OS_OBJECT_TYPE_OS_TASK, 518
 OS_OBJECT_TYPE_OS_TIMEBASE, 518
 OS_OBJECT_TYPE_OS_TIMECB, 518
 OS_OBJECT_TYPE_UNDEFINED, 519
 OS_OBJECT_TYPE_USER, 519
OSAL Printf APIs, 539
 OS_printf, 539
 OS_printf_disable, 539
 OS_printf_enable, 540
OSAL Real Time Clock APIs, 451
 OS_GetLocalTime, 451
 OS_SetLocalTime, 452
 OS_TimeAdd, 453
 OS_TimeAssembleFromMicroseconds, 453
 OS_TimeAssembleFromMilliseconds, 454
 OS_TimeAssembleFromNanoseconds, 454
 OS_TimeAssembleFromSubseconds, 455
 OS_TimeGetFractionalPart, 456
 OS_TimeGetMicrosecondsPart, 456
 OS_TimeGetMillisecondsPart, 457
 OS_TimeGetNanosecondsPart, 458
 OS_TimeGetSubsecondsPart, 459
 OS_TimeGetTotalMicroseconds, 459
 OS_TimeGetTotalMilliseconds, 460
 OS_TimeGetTotalNanoseconds, 460
 OS_TimeGetTotalSeconds, 461
 OS_TimeSubtract, 462
OSAL Reference Point For Seek Offset Defines, 492
 OS_SEEK_CUR, 492
 OS_SEEK_END, 492
 OS_SEEK_SET, 492
OSAL Return Code Defines, 478
 OS_ERR_BAD_ADDRESS, 480
 OS_ERR_FILE, 480
 OS_ERR_INCORRECT_OBJ_STATE, 480
 OS_ERR_INCORRECT_OBJ_TYPE, 481
 OS_ERR_INVALID_ARGUMENT, 481
 OS_ERR_INVALID_ID, 481
 OS_ERR_INVALID_PRIORITY, 481
OSAL Semaphore State Defines, 443
 OS_SEM_EMPTY, 443
 OS_SEM_FULL, 443
OSAL Shell APIs, 551
 OS_ShellOutputToFile, 551
OSAL Socket Address APIs, 552
 OS_SocketAddrFromString, 552
 OS_SocketAddrGetPort, 553
 OS_SocketAddrInit, 553
 OS_SocketAddrSetPort, 554
 OS_SocketAddrToString, 555
OSAL Socket Management APIs, 556
 OS_SocketAccept, 556
OSAL Stream API Defines, 483
 OS_ERR_INVALID_SIZE, 481
 OS_ERR_NAME_NOT_FOUND, 482
 OS_ERR_NAME_TAKEN, 482
 OS_ERR_NAME_TOO_LONG, 482
 OS_ERR_NO_FREE_IDS, 482
 OS_ERR_NOT_IMPLEMENTED, 482
 OS_ERR_OBJECT_IN_USE, 483
 OS_ERR_OPERATION_NOT_SUPPORTED, 483
 OS_ERR_OUTPUT_TOO_LARGE, 483
 OS_ERR_SEM_NOT_FULL, 483
 OS_ERR_STREAM_DISCONNECTED, 483
 OS_ERROR_ADDRESS_MISALIGNED, 484
 OS_ERROR_TIMEOUT, 484
 OS_ERROR, 484
 OS_FS_ERR_DEVICE_NOT_FREE, 484
 OS_FS_ERR_DRIVE_NOT_CREATED, 484
 OS_FS_ERR_NAME_TOO_LONG, 485
 OS_FS_ERR_PATH_INVALID, 485
 OS_FS_ERR_PATH_TOO_LONG, 485
 OS_INVALID_INT_NUM, 485
 OS_INVALID_POINTER, 485
 OS_INVALID_SEM_VALUE, 486
 OS_QUEUE_EMPTY, 486
 OS_QUEUE_FULL, 486
 OS_QUEUE_ID_ERROR, 486
 OS_QUEUE_INVALID_SIZE, 486
 OS_QUEUE_TIMEOUT, 487
 OS_SEM_FAILURE, 487
 OS_SEM_TIMEOUT, 487
 OS_SUCCESS, 487
 OS_TIMER_ERR_INTERNAL, 487
 OS_TIMER_ERR_INVALID_ARGS, 488
 OS_TIMER_ERR_TIMER_ID, 488
 OS_TIMER_ERR_UNAVAILABLE, 488
OSAL Select APIs, 546
 OS_SelectFdAdd, 546
 OS_SelectFdClear, 547
 OS_SelectFdIsSet, 547
 OS_SelectFdZero, 548
 OS_SelectMultiple, 548
 OS_SelectSingle, 549
OSAL Semaphores API Defines, 443
 OS_SEM_EMPTY, 443
 OS_SEM_FULL, 443
OSAL Shell API Defines, 551
 OS_ShellOutputToFile, 551
OSAL Socket Address API Defines, 552
 OS_SocketAddrFromString, 552
 OS_SocketAddrGetPort, 553
 OS_SocketAddrInit, 553
 OS_SocketAddrSetPort, 554
 OS_SocketAddrToString, 555
OSAL Socket Management API Defines, 556
 OS_SocketAccept, 556

OS_SocketBind, 558
 OS_SocketConnect, 559
 OS_SocketGetIdByName, 559
 OS_SocketGetInfo, 560
 OS_SocketOpen, 561
 OS_SocketRecvFrom, 561
 OS_SocketSendTo, 562
 OS_SocketShutdown, 563
 OSAL Standard File APIs, 493
 OS_CloseAllFiles, 495
 OS_CloseFileByName, 495
 OS_FDGetInfo, 496
 OS_FileOpenCheck, 497
 OS_OpenCreate, 499
 OS_TimedRead, 502
 OS_TimedWrite, 503
 OS_chmod, 493
 OS_close, 494
 OS_cp, 496
 OS_lseek, 497
 OS_mv, 498
 OS_read, 500
 OS_remove, 500
 OS_rename, 501
 OS_stat, 502
 OS_write, 504
 OSAL Task APIs, 564
 OS_TaskCreate, 564
 OS_TaskDelay, 565
 OS_TaskDelete, 566
 OS_TaskExit, 566
 OS_TaskFindIdBySystemData, 566
 OS_TaskGetId, 567
 OS_TaskGetIdByName, 567
 OS_TaskGetInfo, 568
 OS_TaskInstallDeleteHandler, 569
 OS_TaskSetPriority, 569
 OSAL Time Base APIs, 571
 OS_TimeBaseCreate, 571
 OS_TimeBaseDelete, 572
 OS_TimeBaseGetFreeRun, 573
 OS_TimeBaseGetIdByName, 574
 OS_TimeBaseGetInfo, 574
 OS_TimeBaseSet, 575
 OSAL Timer APIs, 577
 OS_TimerAdd, 577
 OS_TimerCreate, 579
 OS_TimerDelete, 580
 OS_TimerGetIdByName, 581
 OS_TimerGetInfo, 582
 OS_TimerSet, 583
 OSAL_API_VERSION
 osapi-version.h, 1491
 OSAL_BLOCKCOUNT_C
 common_types.h, 1444
 OSAL_INDEX_C
 common_types.h, 1445
 OSAL_OBJTYPE_C
 common_types.h, 1445
 OSAL_PRIORITY_C
 osapi-task.h, 1484
 OSAL_SIZE_C
 common_types.h, 1445
 OSAL_STACKPTR_C
 osapi-task.h, 1484
 OSAL_STATUS_C
 common_types.h, 1445
 OSAL_TASK_STACK_ALLOCATE
 osapi-task.h, 1484
 OSALMajorVersion
 CFE_ES_HousekeepingTlm_Payload, 610
 OSALMinorVersion
 CFE_ES_HousekeepingTlm_Payload, 610
 OSALMissionRevision
 CFE_ES_HousekeepingTlm_Payload, 610
 OSALRevision
 CFE_ES_HousekeepingTlm_Payload, 611
 object_ids
 OS_FdSet, 829
 ObjectName
 CFE_TBL_FileDef, 730
 ObjectSize
 CFE_TBL_FileDef, 730
 Offset
 CFE_TBL_File_Hdr, 729
 OnEvent
 CFE_FS_FileWriteMetaData, 681
 OneHzAdjust
 CFE_TIME_DiagnosticTlm_Payload, 765
 OneHzDirection
 CFE_TIME_DiagnosticTlm_Payload, 765
 OneTimeAdjust
 CFE_TIME_DiagnosticTlm_Payload, 766
 OneTimeDirection
 CFE_TIME_DiagnosticTlm_Payload, 766
 Opt
 CFE_SB_PipeInfoEntry, 700
 os_dirent_t, 827
 FileName, 828
 os_err_name_t
 osapi-error.h, 1462
 os_finfo_t, 830
 FreeFds, 830
 FreeVolumes, 831
 MaxFds, 831
 MaxVolumes, 831
 os_fstat_t, 831
 FileModeBits, 832

FileSize, 832
 FileTime, 832
osal/docs/src/osal_frontpage.dox, 1442
osal/docs/src/osal_fs.dox, 1442
osal/docs/src/osal_timer.dox, 1442
osal/src/os/inc/common_types.h, 1442
osal/src/os/inc/osapi-binsem.h, 1450
osal/src/os/inc/osapi-bsp.h, 1451
osal/src/os/inc/osapi-clock.h, 1452
osal/src/os/inc/osapi-common.h, 1454
osal/src/os/inc/osapi-constants.h, 1457
osal/src/os/inc/osapi-countsem.h, 1458
osal/src/os/inc/osapi-dir.h, 1459
osal/src/os/inc/osapi-error.h, 1460
osal/src/os/inc/osapi-file.h, 1463
osal/src/os/inc/osapi-filesystems.h, 1467
osal/src/os/inc/osapi-heap.h, 1469
osal/src/os/inc/osapi-idmap.h, 1469
osal/src/os/inc/osapi-macros.h, 1471
osal/src/os/inc/osapi-module.h, 1473
osal/src/os/inc/osapi-mutex.h, 1475
osal/src/os/inc/osapi-network.h, 1476
osal/src/os/inc/osapi-printf.h, 1476
osal/src/os/inc/osapi-queue.h, 1477
osal/src/os/inc/osapi-select.h, 1477
osal/src/os/inc/osapi-shell.h, 1479
osal/src/os/inc/osapi-sockets.h, 1479
osal/src/os/inc/osapi-task.h, 1482
osal/src/os/inc/osapi-timebase.h, 1485
osal/src/os/inc/osapi-timer.h, 1486
osal/src/os/inc/osapi-version.h, 1487
osal/src/os/inc/osapi.h, 1493
osal_blockcount_t
 common_types.h, 1447
osal_id_t
 common_types.h, 1447
osal_index_t
 common_types.h, 1447
osal_objtype_t
 common_types.h, 1448
osal_priority_t
 osapi-task.h, 1484
osal_stackptr_t
 osapi-task.h, 1484
osal_status_t
 common_types.h, 1448
osal_task
 osapi-task.h, 1485
osapi-common.h
 OS_Event_t, 1456
 OS_EventHandler_t, 1455
osapi-constants.h
 OS_CHECK, 1457
 OS_MAX_LOCAL_PATH_LEN, 1457
 OS_OBJECT_CREATOR_ANY, 1457
 OS_OBJECT_ID_UNDEFINED, 1458
 OS_PEND, 1458
osapi-dir.h
 OS_DIRENTRY_NAME, 1459
osapi-error.h
 OS_ERROR_NAME_LENGTH, 1462
 os_err_name_t, 1462
osapi-file.h
 OS_FILESTAT_EXEC, 1465
 OS_FILESTAT_ISDIR, 1465
 OS_FILESTAT_MODE, 1465
 OS_FILESTAT_READ, 1465
 OS_FILESTAT_SIZE, 1465
 OS_FILESTAT_TIME, 1466
 OS_FILESTAT_WRITE, 1466
 OS_file_flag_t, 1467
osapi-filesystems.h
 OS_CHK_ONLY, 1468
 OS_REPAIR, 1468
osapi-idmap.h
 OS_OBJECT_INDEX_MASK, 1471
 OS_OBJECT_TYPE_SHIFT, 1471
osapi-macros.h
 ARGCHECK, 1472
 BUGCHECK, 1472
 BUGREPORT, 1473
 LENGTHCHECK, 1473
osapi-module.h
 OS_MODULE_FLAG_GLOBAL_SYMBOLS, 1474
 OS_MODULE_FLAG_LOCAL_SYMBOLS, 1474
osapi-select.h
 OS_StreamState_t, 1478
osapi-sockets.h
 OS_SOCKADDR_MAX_LEN, 1481
 OS_SocketDomain_t, 1481
 OS_SocketShutdownMode_t, 1481
 OS_SocketType_t, 1482
osapi-task.h
 OS_FP_ENABLED, 1483
 OS_MAX_TASK_PRIORITY, 1483
 OSAL_PRIORITY_C, 1484
 OSAL_STACKPTR_C, 1484
 OSAL_TASK_STACK_ALLOCATE, 1484
 osal_priority_t, 1484
 osal_stackptr_t, 1484
 osal_task, 1485
osapi-timebase.h
 OS_TimerSync_t, 1486
osapi-timer.h
 OS_TimerCallback_t, 1487
osapi-version.h
 OS_BUILD_BASELINE, 1488
 OS_BUILD_NUMBER, 1489

OS_GetBuildNumber, 1491
 OS_GetVersionCodeName, 1491
 OS_GetVersionNumber, 1492
 OS_GetVersionString, 1492
 OS_MAJOR_VERSION, 1489
 OS_MINOR_VERSION, 1489
 OS_MISSION_REV, 1489
 OS_REVISION, 1489
 OS_STR_HELPER, 1490
 OS_STR, 1490
 OS_VERSION_CODENAME, 1490
 OS_VERSION_STRING, 1490
 OS_VERSION, 1490
 OSAL_API_VERSION, 1491

osconfig-example.h

OS_BUFFER_MSG_DEPTH, 993
 OS_BUFFER_SIZE, 993
 OS_FS_DEV_NAME_LEN, 993
 OS_FS_PHYS_NAME_LEN, 993
 OS_FS_VOL_NAME_LEN, 994
 OS_MAX_API_NAME, 994
 OS_MAX_BIN_SEMAPHORES, 994
 OS_MAX_CMD_LEN, 994
 OS_MAX_CONSOLES, 995
 OS_MAX_COUNT_SEMAPHORES, 995
 OS_MAX_FILE_NAME, 995
 OS_MAX_FILE_SYSTEMS, 995
 OS_MAX_MODULES, 996
 OS_MAX_MUTEXES, 996
 OS_MAX_NUM_OPEN_DIRS, 996
 OS_MAX_NUM_OPEN_FILES, 996
 OS_MAX_PATH_LEN, 997
 OS_MAX_QUEUES, 997
 OS_MAX_SYM_LEN, 997
 OS_MAX_TASKS, 997
 OS_MAX_TIMEBASES, 998
 OS_MAX_TIMERS, 998
 OS_MODULE_FILE_EXTENSION, 998
 OS_PRINTF_CONSOLE_NAME, 999
 OS_QUEUE_MAX_DEPTH, 999
 OS_SHELL_CMD_INPUT_FILE_NAME, 999
 OS SOCKADDR_MAX_LEN, 999
 OS UTILITYTASK_PRIORITY, 1000
 OS UTILITYTASK_STACK_SIZE, 1000

OutputPort
 CFE_EVS_HousekeepingTlm_Payload, 665

OwnerAppName
 CFE_TBL_TblRegPacket_Payload, 753

PSPMajorVersion
 CFE_ES_HousekeepingTlm_Payload, 613

PSPMinorVersion
 CFE_ES_HousekeepingTlm_Payload, 614

PSPMissionRevision

CFE_ES_HousekeepingTlm_Payload, 614
 PSPRevision
 CFE_ES_HousekeepingTlm_Payload, 614

PacketID
 CFE_EVS_LongEventTlm_Payload, 669
 CFE_EVS_ShortEventTlm_Payload, 677

Padding
 CFE_EVS_AppTlmData, 657
 HS_SetMaxResetsCmd_t, 821

Parameter
 CFE_TBL_NotifyCmd_Payload, 746

Path
 OS_file_prop_t, 829

Payload
 CFE_ES_AppNameCmd, 593
 CFE_ES_DeleteCDSCmd, 599
 CFE_ES_DumpCDSRegistryCmd, 601
 CFE_ES_FileNameCmd, 602
 CFE_ES_HousekeepingTlm, 604
 CFE_ES_MemStatsTlm, 620
 CFE_ES_OneAppTlm, 622
 CFE_ES_OverWriteSysLogCmd, 623
 CFE_ES_ReloadAppCmd, 627
 CFE_ES_RestartCmd, 628
 CFE_ES_SendMemPoolStatsCmd, 630
 CFE_ES_SetMaxPRCountCmd, 632
 CFE_ES_SetPerfFilterMaskCmd, 634
 CFE_ES_SetPerfTriggerMaskCmd, 636
 CFE_ES_StartApp, 638
 CFE_ES_StartPerfDataCmd, 642
 CFE_ES_StopPerfDataCmd, 644
 CFE_EVS_AppNameBitMaskCmd, 648
 CFE_EVS_AppNameCmd, 650
 CFE_EVS_AppNameEventIDCmd, 652
 CFE_EVS_AppNameEventIDMaskCmd, 654
 CFE_EVS_BitMaskCmd, 659
 CFE_EVS_HousekeepingTlm, 661
 CFE_EVS_LongEventTlm, 668
 CFE_EVS_SetEventFormatModeCmd, 674
 CFE_EVS_SetLogModeCmd, 676
 CFE_EVS_ShortEventTlm, 676
 CFE_EVS_WriteAppDataFileCmd, 678
 CFE_EVS_WriteLogDataFileCmd, 679
 CFE_SB_AllSubscriptionsTlm, 685
 CFE_SB_HousekeepingTlm, 687
 CFE_SB_RouteCmd, 703
 CFE_SB_SingleSubscriptionTlm, 707
 CFE_SB_StatsTlm, 709
 CFE_SB_WriteFileInfoCmd, 717
 CFE_TBL_AbortLoadCmd, 719
 CFE_TBL_ActivateCmd, 721
 CFE_TBL_DeleteCDSCmd, 724
 CFE_TBL_DumpCmd, 725
 CFE_TBL_DumpRegistryCmd, 727

CFE_TBL_HousekeepingTlm, 732
CFE_TBL_LoadCmd, 742
CFE_TBL_NotifyCmd, 745
CFE_TBL_SendRegistryCmd, 747
CFE_TBL_TableRegistryTlm, 749
CFE_TBL_ValidateCmd, 755
CFE_TIME_DiagnosticTlm, 757
CFE_TIME_HousekeepingTlm, 771
CFE_TIME_OneHzAdjustmentCmd, 777
CFE_TIME_SetLeapSecondsCmd, 779
CFE_TIME_SetSignalCmd, 780
CFE_TIME_SetSourceCmd, 781
CFE_TIME_SetStateCmd, 782
CFE_TIME_TimeCmd, 787
CFE_TIME_ToneDataCmd, 789
PeakMemInUse
 CFE_SB_StatsTlm_Payload, 713
PeakMsgIdsInUse
 CFE_SB_StatsTlm_Payload, 713
PeakPipesInUse
 CFE_SB_StatsTlm_Payload, 713
PeakQueueDepth
 CFE_SB_PipeDepthStats, 697
 CFE_SB_PipeInfoEntry, 700
PeakSBBuffersInUse
 CFE_SB_StatsTlm_Payload, 713
PeakSubscriptionsInUse
 CFE_SB_StatsTlm_Payload, 714
PerfDataCount
 CFE_ES_HousekeepingTlm_Payload, 611
PerfDataEnd
 CFE_ES_HousekeepingTlm_Payload, 611
PerfDataStart
 CFE_ES_HousekeepingTlm_Payload, 611
PerfDataToWrite
 CFE_ES_HousekeepingTlm_Payload, 612
PerfFilterMask
 CFE_ES_HousekeepingTlm_Payload, 612
PerfMode
 CFE_ES_HousekeepingTlm_Payload, 612
PerfState
 CFE_ES_HousekeepingTlm_Payload, 612
PerfTriggerCount
 CFE_ES_HousekeepingTlm_Payload, 613
PerfTriggerMask
 CFE_ES_HousekeepingTlm_Payload, 613
Pipe
 CFE_SB_RouteCmd_Payload, 704
 CFE_SB_SingleSubscriptionTlm_Payload, 708
 CFE_SB_SubEntries, 716
PipeDepthStats
 CFE_SB_StatsTlm_Payload, 714
Pipeld
 CFE_SB_PipeDepthStats, 698
CFE_SB_PipeInfoEntry, 700
CFE_SB_RoutingFileEntry, 706
PipeName
 CFE_SB_PipeInfoEntry, 700
 CFE_SB_RoutingFileEntry, 706
PipeOptsErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 692
PipeOverflowErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 692
PipesInUse
 CFE_SB_StatsTlm_Payload, 714
PktSegment
 CFE_SB_AllSubscriptionsTlm_Payload, 686
PoolHandle
 CFE_ES_PoolStatsTlm_Payload, 626
 CFE_ES_SendMemPoolStatsCmd_Payload, 631
PoolSize
 CFE_ES_MemPoolStats, 619
PoolStats
 CFE_ES_PoolStatsTlm_Payload, 626
Priority
 CFE_ES_AppInfo, 591
 CFE_ES_StartAppCmd_Payload, 640
 CFE_ES_TaskInfo, 645
 CFE_SB_Qos_t, 702
priority
 OS_task_prop_t, 845
ProcessorID
 CFE_EVS_PacketID, 671
 CFE_FS_Header, 683
ProcessorResets
 CFE_ES_HousekeepingTlm_Payload, 613
psp/fsw/inc/cfe_psp.h, 1493
Ptr
 CFE_ES_PoolAlign, 625
Qos
 CFE_SB_SingleSubscriptionTlm_Payload, 708
 CFE_SB_SubEntries, 716
RegisteredCoreApps
 CFE_ES_HousekeepingTlm_Payload, 614
RegisteredExternalApps
 CFE_ES_HousekeepingTlm_Payload, 615
RegisteredLibs
 CFE_ES_HousekeepingTlm_Payload, 615
RegisteredTasks
 CFE_ES_HousekeepingTlm_Payload, 615
Reliability
 CFE_SB_Qos_t, 702
Reserved
 CFE_TBL_File_Hdr, 729
ResetSubtype
 CFE_ES_HousekeepingTlm_Payload, 615
ResetType

CFE_ES_HousekeepingTlm_Payload, [616](#)
 ResetsPerformed
 HS_CDSData_t, [805](#)
 HS_HkPacket_t, [815](#)
 ResetsPerformedNot
 HS_CDSData_t, [805](#)
 ResourceId
 CFE_ES_AppInfo, [591](#)
 ResourceName
 HS_XCTEntry_t, [825](#)
 ResourceType
 HS_XCTEntry_t, [825](#)
 RestartType
 CFE_ES_RestartCmd_Payload, [629](#)
 RunStatus
 HS_AppData_t, [802](#)

 SBBuffersInUse
 CFE_SB_StatsTlm_Payload, [714](#)
 SIZE_BYTE
 cfe_psp.h, [1505](#)
 SIZE_HALF
 cfe_psp.h, [1505](#)
 SIZE_WORD
 cfe_psp.h, [1505](#)
 sample_mission_cfg.h
 CFE_MISSION_ES_APP_TLM_MSG, [1068](#)
 CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN,
 [1068](#)
 CFE_MISSION_ES_CDS_MAX_NAME_LENGTH,
 [1069](#)
 CFE_MISSION_ES_CMD_MSG, [1069](#)
 CFE_MISSION_ES_CRC_16, [1070](#)
 CFE_MISSION_ES_CRC_32, [1070](#)
 CFE_MISSION_ES_CRC_8, [1070](#)
 CFE_MISSION_ES_DEFAULT_CRC, [1070](#)
 CFE_MISSION_ES_HK_TLM_MSG, [1070](#)
 CFE_MISSION_ES_MAX_APPLICATIONS, [1071](#)
 CFE_MISSION_ES_MEMSTATS_TLM_MSG, [1071](#)
 CFE_MISSION_ES_PERF_MAX_IDS, [1071](#)
 CFE_MISSION_ES_POOL_MAX_BUCKETS, [1072](#)
 CFE_MISSION_ES_SEND_HK_MSG, [1072](#)
 CFE_MISSION_EVS_CMD_MSG, [1073](#)
 CFE_MISSION_EVS_HK_TLM_MSG, [1073](#)
 CFE_MISSION_EVS_LONG_EVENT_MSG_MSG,
 [1073](#)
 CFE_MISSION_EVS_MAX_MESSAGE_LENGTH,
 [1073](#)
 CFE_MISSION_EVS_SEND_HK_MSG, [1074](#)
 CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG,
 [1074](#)
 CFE_MISSION_MAX_API_LEN, [1074](#)
 CFE_MISSION_MAX_FILE_LEN, [1074](#)
 CFE_MISSION_MAX_PATH_LEN, [1075](#)

 CFE_MISSION_SB_ALLSUBS_TLM_MSG, [1075](#)
 CFE_MISSION_SB_CMD_MSG, [1076](#)
 CFE_MISSION_SB_HK_TLM_MSG, [1076](#)
 CFE_MISSION_SB_MAX_PIPES, [1076](#)
 CFE_MISSION_SB_MAX_SB_MSG_SIZE, [1076](#)
 CFE_MISSION_SB_ONESUB_TLM_MSG, [1077](#)
 CFE_MISSION_SB_SEND_HK_MSG, [1077](#)
 CFE_MISSION_SB_STATS_TLM_MSG, [1077](#)
 CFE_MISSION_SB_SUB_RPT_CTRL_MSG, [1077](#)
 CFE_MISSION_TBL_CMD_MSG, [1078](#)
 CFE_MISSION_TBL_HK_TLM_MSG, [1078](#)
 CFE_MISSION_TBL_MAX_FULL_NAME_LEN,
 [1078](#)
 CFE_MISSION_TBL_MAX_NAME_LENGTH, [1078](#)
 CFE_MISSION_TBL_REG_TLM_MSG, [1079](#)
 CFE_MISSION_TBL_SEND_HK_MSG, [1079](#)
 CFE_MISSION_TEST_CMD_MSG, [1079](#)
 CFE_MISSION_TEST_HK_TLM_MSG, [1079](#)
 CFE_MISSION_TIME_1HZ_CMD_MSG, [1080](#)
 CFE_MISSION_TIME_AT_TONE_WAS, [1080](#)
 CFE_MISSION_TIME_AT_TONE_WILL_BE, [1080](#)
 CFE_MISSION_TIME_CFG_DEFAULT_TAI, [1080](#)
 CFE_MISSION_TIME_CFG_DEFAULT_UTC, [1081](#)
 CFE_MISSION_TIME_CFG_FAKE_TONE, [1081](#)
 CFE_MISSION_TIME_CMD_MSG, [1081](#)
 CFE_MISSION_TIME_DATA_CMD_MSG, [1082](#)
 CFE_MISSION_TIME_DEF_DELAY_SECS, [1082](#)
 CFE_MISSION_TIME_DEF_DELAY_SUBS, [1082](#)
 CFE_MISSION_TIME_DEF_LEAPS, [1082](#)
 CFE_MISSION_TIME_DEF_MET_SECS, [1082](#)
 CFE_MISSION_TIME_DEF_MET_SUBS, [1083](#)
 CFE_MISSION_TIME_DEF_STCF_SECS, [1083](#)
 CFE_MISSION_TIME_DEF_STCF_SUBS, [1083](#)
 CFE_MISSION_TIME_DIAG_TLM_MSG, [1083](#)
 CFE_MISSION_TIME_EPOCH_DAY, [1084](#)
 CFE_MISSION_TIME_EPOCH_HOUR, [1084](#)
 CFE_MISSION_TIME_EPOCH_MICROS, [1084](#)
 CFE_MISSION_TIME_EPOCH_MINUTE, [1084](#)
 CFE_MISSION_TIME_EPOCH_SECOND, [1084](#)
 CFE_MISSION_TIME_EPOCH_YEAR, [1084](#)
 CFE_MISSION_TIME_FS_FACTOR, [1085](#)
 CFE_MISSION_TIME_HK_TLM_MSG, [1085](#)
 CFE_MISSION_TIME_MAX_ELAPSED, [1086](#)
 CFE_MISSION_TIME_MIN_ELAPSED, [1086](#)
 CFE_MISSION_TIME_SEND_CMD_MSG, [1086](#)
 CFE_MISSION_TIME_SEND_HK_MSG, [1086](#)
 CFE_MISSION_TIME_TONE_CMD_MSG, [1087](#)

 sample_perfids.h
 CFE_MISSION_ES_MAIN_PERF_ID, [1088](#)
 CFE_MISSION_ES_PERF_EXIT_BIT, [1088](#)
 CFE_MISSION_EVS_MAIN_PERF_ID, [1088](#)
 CFE_MISSION_SB_MAIN_PERF_ID, [1088](#)
 CFE_MISSION_SB_MSG_LIM_PERF_ID, [1088](#)
 CFE_MISSION_SB_PIPE_OFLOW_PERF_ID, [1089](#)

CFE_MISSION_TBL_MAIN_PERF_ID, 1089
CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID, 1089
CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID, 1089
CFE_MISSION_TIME_MAIN_PERF_ID, 1089
CFE_MISSION_TIME_SENDMET_PERF_ID, 1090
CFE_MISSION_TIME_TONE1HZISR_PERF_ID, 1090
CFE_MISSION_TIME_TONE1HZTASK_PERF_ID, 1090
Seconds
 CFE_TIME_OneHzAdjustmentCmd_Payload, 778
 CFE_TIME_SysTime, 786
 CFE_TIME_TimeCmd_Payload, 788
Seconds1HzAdj
 CFE_TIME_HousekeepingTlm_Payload, 773
SecondsDelay
 CFE_TIME_HousekeepingTlm_Payload, 773
SecondsMET
 CFE_TIME_HousekeepingTlm_Payload, 774
SecondsSTCF
 CFE_TIME_HousekeepingTlm_Payload, 774
SendErrors
 CFE_SB_PipeInfoEntry, 701
Sequence
 CCSDS_PrimaryHeader, 585
ServerFlyState
 CFE_TIME_DiagnosticTlm_Payload, 766
ServiceWatchdogFlag
 HS_AppData_t, 802
Size
 CFE_ES_CDSRegDumpRec, 598
 CFE_TBL_Info, 741
 CFE_TBL_TblRegPacket_Payload, 753
SpacecraftID
 CFE_EVS_PacketID, 672
 CFE_FS_Header, 683
Spare
 CFE_ES_TaskInfo, 646
 CFE_EVS_AppNameBitMaskCmd_Payload, 649
 CFE_EVS_BitMaskCmd_Payload, 660
 CFE_EVS_SetEventFormatCode_Payload, 673
 CFE_EVS_SetLogMode_Payload, 675
 CFE_SB_PipeDepthStats, 698
 CFE_SB_PipeInfoEntry, 701
 CFE_SB_RouteCmd_Payload, 704
Spare1
 CFE_EVS_HousekeepingTlm_Payload, 665
 CFE_EVS_LongEventTlm_Payload, 669
Spare2
 CFE_EVS_HousekeepingTlm_Payload, 665
 CFE_EVS_LongEventTlm_Payload, 669
Spare2Align
 CFE_SB_HousekeepingTlm_Payload, 692
Spare3
 CFE_EVS_HousekeepingTlm_Payload, 666
SpareBytes
 HS_AppData_t, 802
 HS_HkPacket_t, 815
stack_size
 OS_task_prop_t, 845
StackSize
 CFE_ES_AppInfo, 591
 CFE_ES_StartAppCmd_Payload, 640
 CFE_ES_TaskInfo, 646
start_time
 OS_timer_prop_t, 849
StartAddress
 CFE_ES_AppInfo, 592
State
 CFE_SB_RoutingFileEntry, 706
StatusFlags
 HS_HkPacket_t, 816
StreamId
 CCSDS_PrimaryHeader, 585
SubType
 CFE_FS_Header, 683
 CFE_SB_SingleSubscriptionTlm_Payload, 709
SubscribeErrorCounter
 CFE_SB_HousekeepingTlm_Payload, 692
SubscriptionsInUse
 CFE_SB_StatsTlm_Payload, 715
Subseconds
 CFE_TIME_OneHzAdjustmentCmd_Payload, 778
 CFE_TIME_SysTime, 786
Subsecs1HzAdj
 CFE_TIME_HousekeepingTlm_Payload, 774
SubsecsDelay
 CFE_TIME_HousekeepingTlm_Payload, 774
SubsecsMET
 CFE_TIME_HousekeepingTlm_Payload, 775
SubsecsSTCF
 CFE_TIME_HousekeepingTlm_Payload, 775
Subsystem
 CCSDS_ExtendedHeader, 584
SuccessValCounter
 CFE_TBL_HousekeepingTlm_Payload, 738
SysLogBytesUsed
 CFE_ES_HousekeepingTlm_Payload, 616
SysLogEntries
 CFE_ES_HousekeepingTlm_Payload, 616
SysLogMode
 CFE_ES_HousekeepingTlm_Payload, 616
SysLogSize
 CFE_ES_HousekeepingTlm_Payload, 617
SystemId
 CCSDS_ExtendedHeader, 584

Table
 CFE_ES_CDSRegDumpRec, 598

TableLoadedOnce
 CFE_TBL_Info, 741
 CFE_TBL_TblRegPacket_Payload, 754

TableName
 CFE_TBL_AbortLoadCmd_Payload, 720
 CFE_TBL_ActivateCmd_Payload, 722
 CFE_TBL_DelCDSCmd_Payload, 723
 CFE_TBL_DumpCmd_Payload, 726
 CFE_TBL_File_Hdr, 729
 CFE_TBL_FileDef, 731
 CFE_TBL_SendRegistryCmd_Payload, 748
 CFE_TBL_ValidateCmd_Payload, 756

TaskId
 CFE_ES_TaskInfo, 646

TaskName
 CFE_ES_TaskInfo, 646

TelemetryHeader
 CFE_ES_HousekeepingTlm, 604
 CFE_ES_MemStatsTlm, 620
 CFE_ES_OneAppTlm, 622
 CFE_EVS_HousekeepingTlm, 661
 CFE_EVS_LongEventTlm, 668
 CFE_EVS_ShortEventTlm, 677
 CFE_SB_AllSubscriptionsTlm, 685
 CFE_SB_HousekeepingTlm, 687
 CFE_SB_SingleSubscriptionTlm, 707
 CFE_SB_StatsTlm, 710
 CFE_TBL_HousekeepingTlm, 732
 CFE_TBL_TableRegistryTlm, 749
 CFE_TIME_DiagnosticTlm, 757
 CFE_TIME_HousekeepingTlm, 771

TgtFilename
 CFE_TBL_FileDef, 731

ThisIdleTaskExec
 HS_CustomData_t, 807

ticks
 OS_time_t, 846

TimeOfLastUpdate
 CFE_TBL_Info, 741
 CFE_TBL_TblRegPacket_Payload, 754

TimeSeconds
 CFE_FS_Header, 684

TimeSinceTone
 CFE_TIME_DiagnosticTlm_Payload, 766

TimeSource
 CFE_TIME_SourceCmd_Payload, 784

TimeSubSeconds
 CFE_FS_Header, 684

TlmHeader
 HS_HkPacket_t, 816

ToneDataCounter
 CFE_TIME_DiagnosticTlm_Payload, 767

ToneDataLatch
 CFE_TIME_DiagnosticTlm_Payload, 767

ToneIntCounter
 CFE_TIME_DiagnosticTlm_Payload, 767

ToneIntErrorCounter
 CFE_TIME_DiagnosticTlm_Payload, 767

ToneMatchCounter
 CFE_TIME_DiagnosticTlm_Payload, 768

ToneMatchErrorCounter
 CFE_TIME_DiagnosticTlm_Payload, 768

ToneOverLimit
 CFE_TIME_DiagnosticTlm_Payload, 768

ToneSignalCounter
 CFE_TIME_DiagnosticTlm_Payload, 768

ToneSignalLatch
 CFE_TIME_DiagnosticTlm_Payload, 769

ToneSource
 CFE_TIME_SignalCmd_Payload, 783

ToneTaskCounter
 CFE_TIME_DiagnosticTlm_Payload, 769

ToneUnderLimit
 CFE_TIME_DiagnosticTlm_Payload, 769

total_blocks
 OS_statvfs_t, 844

TotalSegments
 CFE_SB_AllSubscriptionsTlm_Payload, 686

TriggerMask
 CFE_ES_SetPerfTrigMaskCmd_Payload, 637

TriggerMaskNum
 CFE_ES_SetPerfTrigMaskCmd_Payload, 637

TriggerMode
 CFE_ES_StartPerfCmd_Payload, 641

Type
 CFE_ES_AppInfo, 592

uint16
 common_types.h, 1448

uint32
 common_types.h, 1448

uint64
 common_types.h, 1448

uint8
 common_types.h, 1449

UnmarkedMem
 CFE_SB_HousekeepingTlm_Payload, 693

UnregisteredAppCounter
 CFE_EVS_HousekeepingTlm_Payload, 666

User
 OS_file_prop_t, 830

UserDefAddr
 CFE_TBL_Info, 741

UtilArray
 HS_CustomData_t, 808

UtilArrayIndex

HS_CustomData_t, 808
UtilArrayMask
 HS_CustomData_t, 808
UtilCallsPerMark
 HS_CustomData_t, 808
UtilCpuAvg
 HS_AppData_t, 803
 HS_HkPacket_t, 816
UtilCpuPeak
 HS_AppData_t, 803
 HS_HkPacket_t, 816
UtilCycleCounter
 HS_CustomData_t, 809
UtilDiv
 HS_CustomData_t, 809
UtilMask
 HS_CustomData_t, 809
UtilMult1
 HS_CustomData_t, 809
UtilMult2
 HS_CustomData_t, 810
UtilizationTracker
 HS_AppData_t, 803

valid
 OS_module_address_t, 835
ValidationCounter
 CFE_TBL_HousekeepingTlm_Payload, 738
ValidationFuncPtr
 CFE_TBL_TblRegPacket_Payload, 754
Value
 CFE_SB_MsgId_t, 695
value
 OS_bin_sem_prop_t, 826
 OS_count_sem_prop_t, 827
VersionCounter
 CFE_TIME_DiagnosticTlm_Payload, 769
VirtualMET
 CFE_TIME_DiagnosticTlm_Payload, 770

WakeupPipe
 HS_AppData_t, 803