FM User's Guide

Generated by Doxygen 1.8.13

Contents

1 CFS File Manager (FM) Documentation **2**

1.1 CFS File Manager Introduction . . . 2

1.2 CFS File Manager Overview . . . 3

1.3 CFS File Manager Operations . . . 4

1.4 CFS File Manager Commands . . . 5

1.5 CFS File Manager Telemetry . . . 5

1.6 CFS File Manager Events . . . 6

1.7 CFS File Manager Deployment Guide . . . 6

1.8 CFS File Manager Configuration . . . 6

1.9 CFS File Manager Table Definitions . . . 6

1.10 CFS File Manager Operational Constraints . . . 6

1.11 CFS File Manager Frequently Asked Questions . . . 6

2 Core Flight Executive Documentation **7**

2.1 Background . . . 8

2.2 Applicable Documents . . . 9

2.3 Version Numbers . . . 10

2.4 Dependencies . . . 11

2.5 Acronyms . . . 11

2.6 cFE Executive Services Overview . . . 13

2.6.1 Terminology . . . 14

2.6.2 Software Reset . . . 16

2.6.3 Reset Types and Subtypes . . . 16

2.6.4 Exception and Reset (ER) Log . . . 17

2.6.5 Application and Child Task Management . . . 17

2.6.6 Starting an Application . . . 17

2.6.7 Stopping an Application . . . 18

2.6.8 Restarting an Application . . . 18
2.6.9 Reloading an Application . . . 18
2.6.10 Listing Current Applications . . . 19
2.6.11 Listing Current Tasks . . . 20
2.6.12 Loading Common Libraries . . . 20
2.6.13 Basic File System . . . 20
2.6.14 Performance Data Collection . . . 21
2.6.15 Critical Data Store . . . 22
2.6.16 Memory Pool . . . 23
2.6.17 System Log . . . 25
2.6.18 Version Identification . . . 25
2.6.19 Executive Services Frequently Asked Questions . . . 26
2.7 cFE Executive Services Commands . . . 26
2.8 cFE Executive Services Telemetry . . . 27
2.9 cFE Executive Services Configuration Parameters . . . 28
2.10 cFE Event Services Overview . . . 31
2.10.1 Event Message Format . . . 32
2.10.2 Local Event Log . . . 33
2.10.3 Event Message Control . . . 33
2.10.4 Event Message Filtering . . . 35
2.10.5 EVS Registry . . . 35
2.10.6 EVS Counters . . . 36
2.10.7 Resetting EVS Counters . . . 37
2.10.8 Effects of a Processor Reset on EVS . . . 37
2.10.9 Frequently Asked Questions about Event Services . . . 37
2.11 cFE Event Services Commands . . . 38
2.12 cFE Event Services Telemetry . . . 40
2.13 cFE Event Services Configuration Parameters . . . 40

2.14 cFE Software Bus Overview . . . 41
2.14.1 Software Bus Terminology . . . 41
2.14.2 Autonomous Actions . . . 44
2.14.3 Operation of the SB Software . . . 44
2.14.4 Frequently Asked Questions about Software Bus . . . 48
2.15 cFE Software Bus Commands . . . 50
2.16 cFE Software Bus Telemetry . . . 51
2.17 cFE Software Bus Configuration Parameters . . . 51
2.18 cFE Table Services Overview . . . 52
2.18.1 Managing Tables . . . 53
2.18.2 cFE Table Types and Table Options . . . 54
2.18.3 Table Registry . . . 56
2.18.4 Table Services Telemetry . . . 57
2.18.5 Effects of Processor Reset on Tables . . . 57
2.18.6 Frequently Asked Questions about Table Services . . . 57
2.19 cFE Table Services Commands . . . 59
2.20 cFE Table Services Telemetry . . . 60
2.21 cFE Table Services Configuration Parameters . . . 60
2.22 cFE Time Services Overview . . . 61
2.22.1 Time Components . . . 63
2.22.2 Time Structure . . . 64
2.22.3 Time Formats . . . 64
2.22.4 Time Configuration . . . 65
2.22.5 Time Format Selection . . . 70
2.22.6 Enabling Fake Tone Signal . . . 71
2.22.7 Selecting Tone and Data Ordering . . . 71
2.22.8 Specifying Tone and Data Window . . . 72
2.22.9 Specifying Time Server/Client . . . 72

2.22.10 Specifying Time Tone Byte Order . . . 73
2.22.11 Virtual MET . . . 73
2.22.12 Specifying Time Source . . . 73
2.22.13 Specifying Time Signal . . . 74
2.22.14 Time Services Paradigm(s) . . . 75
2.22.15 Flywheeling . . . 75
2.22.16 Time State . . . 76
2.22.17 Initialization . . . 76
2.22.18 Power-On Reset . . . 77
2.22.19 Processor Reset . . . 78
2.22.20 Initialization . . . 78
2.22.21 Power-On Reset . . . 79
2.22.22 Processor Reset . . . 80
2.22.23 Normal Operation . . . 80
2.22.24 Client . . . 82
2.22.25 Server . . . 83
2.22.26 Setting Time . . . 84
2.22.27 Adjusting Time . . . 85
2.22.28 Setting MET . . . 85
2.22.29 Frequently Asked Questions . . . 85
2.23 cFE Time Services Commands . . . 86
2.24 cFE Time Services Telemetry . . . 87
2.25 cFE Time Services Configuration Parameters . . . 87
2.26 cFE Event Message Cross Reference . . . 88
2.27 cFE Command Mnemonic Cross Reference . . . 88
2.28 cFE Telemetry Mnemonic Cross Reference . . . 92

3 Glossary of Terms **103**

4 cFE Application Programmer's Interface (API) Reference **104**

5 Osal API Documentation **111**

5.1 OSAL Introduction 113

5.2 File System Overview 113

5.3 File Descriptors In Osal 114

5.4 Timer Overview 115

6 cFE Mission Configuration Parameters **115**

7 Module Index **115**

7.1 Modules 115

8 Data Structure Index **118**

8.1 Data Structures 118

9 File Index **127**

9.1 File List 128

10 Module Documentation **131**

10.1 CFS File Manager Mission Configuration 131

10.1.1 Detailed Description 131

10.1.2 Macro Definition Documentation 131

10.2 CFS File Manager Command Message IDs 132

10.2.1 Detailed Description 132

10.2.2 Macro Definition Documentation 132

10.3 CFS File Manager Telemetry Message IDs 133

10.3.1 Detailed Description 133

10.3.2 Macro Definition Documentation 133

10.4 CFS File Manager Platform Configuration 134

10.4.1 Detailed Description 135

10.4.2 Macro Definition Documentation . . . 135
10.5 CFS File Manager Event IDs . . . 146
10.5.1 Detailed Description . . . 158
10.5.2 Macro Definition Documentation . . . 158
10.6 CFS File Manager Command Structures . . . 276
10.6.1 Detailed Description . . . 276
10.7 CFS File Manager Telemetry . . . 277
10.7.1 Detailed Description . . . 277
10.8 CFS File Manager Command Codes . . . 278
10.8.1 Detailed Description . . . 278
10.8.2 Macro Definition Documentation . . . 278
10.9 CFS File Manager Version . . . 301
10.9.1 Detailed Description . . . 301
10.9.2 Macro Definition Documentation . . . 301
10.10 cFE Return Code Defines . . . 302
10.10.1 Detailed Description . . . 307
10.10.2 Macro Definition Documentation . . . 307
10.11 cFE Resource ID APIs . . . 339
10.11.1 Detailed Description . . . 339
10.11.2 Function Documentation . . . 339
10.12 cFE Entry/Exit APIs . . . 343
10.12.1 Detailed Description . . . 343
10.12.2 Function Documentation . . . 343
10.13 cFE Application Control APIs . . . 345
10.13.1 Detailed Description . . . 345
10.13.2 Function Documentation . . . 345
10.14 cFE Application Behavior APIs . . . 348
10.14.1 Detailed Description . . . 348

10.14.2 Function Documentation . . . 348
10.15cFE Information APIs . . . 353
10.15.1 Detailed Description . . . 353
10.15.2 Function Documentation . . . 353
10.16cFE Child Task APIs . . . 363
10.16.1 Detailed Description . . . 363
10.16.2 Function Documentation . . . 363
10.17cFE Miscellaneous APIs . . . 368
10.17.1 Detailed Description . . . 368
10.17.2 Function Documentation . . . 368
10.18cFE Critical Data Store APIs . . . 372
10.18.1 Detailed Description . . . 372
10.18.2 Function Documentation . . . 372
10.19cFE Memory Manager APIs . . . 377
10.19.1 Detailed Description . . . 377
10.19.2 Function Documentation . . . 377
10.20cFE Performance Monitor APIs . . . 385
10.20.1 Detailed Description . . . 385
10.20.2 Macro Definition Documentation . . . 385
10.20.3 Function Documentation . . . 386
10.21cFE Generic Counter APIs . . . 388
10.21.1 Detailed Description . . . 388
10.21.2 Function Documentation . . . 388
10.22cFE Registration APIs . . . 396
10.22.1 Detailed Description . . . 396
10.22.2 Function Documentation . . . 396
10.23cFE Send Event APIs . . . 398
10.23.1 Detailed Description . . . 398

10.23.2 Function Documentation 398
10.24cFE Reset Event Filter APIs 403
10.24.1 Detailed Description 403
10.24.2 Function Documentation 403
10.25cFE File Header Management APIs 405
10.25.1 Detailed Description 405
10.25.2 Function Documentation 405
10.26cFE File Utility APIs 410
10.26.1 Detailed Description 410
10.26.2 Function Documentation 410
10.27cFE Generic Message APIs 416
10.27.1 Detailed Description 416
10.27.2 Function Documentation 416
10.28cFE Message Primary Header APIs 417
10.28.1 Detailed Description 417
10.28.2 Function Documentation 417
10.29cFE Message Extended Header APIs 428
10.29.1 Detailed Description 428
10.29.2 Function Documentation 428
10.30cFE Message Secondary Header APIs 435
10.30.1 Detailed Description 435
10.30.2 Function Documentation 435
10.31cFE Message Id APIs 441
10.31.1 Detailed Description 441
10.31.2 Function Documentation 441
10.32cFE Pipe Management APIs 444
10.32.1 Detailed Description 444
10.32.2 Function Documentation 444

10.33cFE Message Subscription Control APIs . . . 451
10.33.1 Detailed Description . . . 451
10.33.2 Function Documentation . . . 451
10.34cFE Send/Receive Message APIs . . . 457
10.34.1 Detailed Description . . . 457
10.34.2 Function Documentation . . . 457
10.35cFE Zero Copy APIs . . . 460
10.35.1 Detailed Description . . . 460
10.35.2 Function Documentation . . . 460
10.36cFE Message Characteristics APIs . . . 463
10.36.1 Detailed Description . . . 463
10.36.2 Function Documentation . . . 463
10.37cFE Message ID APIs . . . 468
10.37.1 Detailed Description . . . 468
10.37.2 Function Documentation . . . 468
10.38cFE SB Pipe options . . . 471
10.38.1 Detailed Description . . . 471
10.38.2 Macro Definition Documentation . . . 471
10.39cFE Registration APIs . . . 472
10.39.1 Detailed Description . . . 472
10.39.2 Function Documentation . . . 472
10.40cFE Manage Table Content APIs . . . 478
10.40.1 Detailed Description . . . 478
10.40.2 Function Documentation . . . 478
10.41cFE Access Table Content APIs . . . 485
10.41.1 Detailed Description . . . 485
10.41.2 Function Documentation . . . 485
10.42cFE Get Table Information APIs . . . 490

10.42.1 Detailed Description 490
10.42.2 Function Documentation 490
10.43cFE Table Type Defines 494
10.43.1 Detailed Description 494
10.43.2 Macro Definition Documentation 494
10.44cFE Get Current Time APIs 498
10.44.1 Detailed Description 498
10.44.2 Function Documentation 498
10.45cFE Get Time Information APIs 502
10.45.1 Detailed Description 502
10.45.2 Function Documentation 502
10.46cFE Time Arithmetic APIs 505
10.46.1 Detailed Description 505
10.46.2 Function Documentation 505
10.47cFE Time Conversion APIs 508
10.47.1 Detailed Description 508
10.47.2 Function Documentation 508
10.48cFE External Time Source APIs 511
10.48.1 Detailed Description 511
10.48.2 Function Documentation 511
10.49cFE Miscellaneous Time APIs 516
10.49.1 Detailed Description 516
10.49.2 Function Documentation 516
10.50cFE Resource ID base values 518
10.50.1 Detailed Description 518
10.50.2 Enumeration Type Documentation 518
10.51cFE Clock State Flag Defines 520
10.51.1 Detailed Description 520

10.51.2 Macro Definition Documentation . . . 520
10.52 OSAL Semaphore State Defines . . . 523
10.52.1 Detailed Description . . . 523
10.52.2 Macro Definition Documentation . . . 523
10.53 OSAL Binary Semaphore APIs . . . 524
10.53.1 Detailed Description . . . 524
10.53.2 Function Documentation . . . 524
10.54 OSAL BSP low level access APIs . . . 530
10.54.1 Detailed Description . . . 530
10.54.2 Function Documentation . . . 530
10.55 OSAL Real Time Clock APIs . . . 531
10.55.1 Detailed Description . . . 531
10.55.2 Function Documentation . . . 531
10.56 OSAL Core Operation APIs . . . 543
10.56.1 Detailed Description . . . 543
10.56.2 Function Documentation . . . 543
10.57 OSAL Counting Semaphore APIs . . . 547
10.57.1 Detailed Description . . . 547
10.57.2 Function Documentation . . . 547
10.58 OSAL Directory APIs . . . 553
10.58.1 Detailed Description . . . 553
10.58.2 Function Documentation . . . 553
10.59 OSAL Return Code Defines . . . 558
10.59.1 Detailed Description . . . 559
10.59.2 Macro Definition Documentation . . . 560
10.60 OSAL Error Info APIs . . . 569
10.60.1 Detailed Description . . . 569
10.60.2 Function Documentation . . . 569

10.61 OSAL File Access Option Defines . . . 571
10.61.1 Detailed Description . . . 571
10.61.2 Macro Definition Documentation . . . 571
10.62 OSAL Reference Point For Seek Offset Defines . . . 572
10.62.1 Detailed Description . . . 572
10.62.2 Macro Definition Documentation . . . 572
10.63 OSAL Standard File APIs . . . 573
10.63.1 Detailed Description . . . 573
10.63.2 Function Documentation . . . 573
10.64 OSAL File System Level APIs . . . 586
10.64.1 Detailed Description . . . 586
10.64.2 Function Documentation . . . 586
10.65 OSAL Heap APIs . . . 595
10.65.1 Detailed Description . . . 595
10.65.2 Function Documentation . . . 595
10.66 OSAL Object Type Defines . . . 596
10.66.1 Detailed Description . . . 596
10.66.2 Macro Definition Documentation . . . 596
10.67 OSAL Object ID Utility APIs . . . 600
10.67.1 Detailed Description . . . 600
10.67.2 Function Documentation . . . 600
10.68 OSAL Dynamic Loader and Symbol APIs . . . 607
10.68.1 Detailed Description . . . 607
10.68.2 Function Documentation . . . 607
10.69 OSAL Mutex APIs . . . 612
10.69.1 Detailed Description . . . 612
10.69.2 Function Documentation . . . 612
10.70 OSAL Network ID APIs . . . 617

10.70.1 Detailed Description 617
10.70.2 Function Documentation 617
10.71 OSAL Printf APIs 619
10.71.1 Detailed Description 619
10.71.2 Function Documentation 619
10.72 OSAL Message Queue APIs 621
10.72.1 Detailed Description 621
10.72.2 Function Documentation 621
10.73 OSAL Select APIs 626
10.73.1 Detailed Description 626
10.73.2 Function Documentation 626
10.74 OSAL Shell APIs 631
10.74.1 Detailed Description 631
10.74.2 Function Documentation 631
10.75 OSAL Socket Address APIs 632
10.75.1 Detailed Description 632
10.75.2 Function Documentation 632
10.76 OSAL Socket Management APIs 636
10.76.1 Detailed Description 636
10.76.2 Function Documentation 636
10.77 OSAL Task APIs 644
10.77.1 Detailed Description 644
10.77.2 Function Documentation 644
10.78 OSAL Time Base APIs 651
10.78.1 Detailed Description 651
10.78.2 Function Documentation 651
10.79 OSAL Timer APIs 657
10.79.1 Detailed Description 657
10.79.2 Function Documentation 657

11 Data Structure Documentation **664**

11.1 CCSDS_ExtendedHeader Struct Reference . . . 664

11.1.1 Detailed Description . . . 664

11.1.2 Field Documentation . . . 664

11.2 CCSDS_PrimaryHeader Struct Reference . . . 665

11.2.1 Detailed Description . . . 665

11.2.2 Field Documentation . . . 665

11.3 CFE_ES_AppInfo Struct Reference . . . 666

11.3.1 Detailed Description . . . 667

11.3.2 Field Documentation . . . 667

11.4 CFE_ES_AppNameCmd Struct Reference . . . 672

11.4.1 Detailed Description . . . 673

11.4.2 Field Documentation . . . 673

11.5 CFE_ES_AppNameCmd_Payload Struct Reference . . . 673

11.5.1 Detailed Description . . . 674

11.5.2 Field Documentation . . . 674

11.6 CFE_ES_AppReloadCmd_Payload Struct Reference . . . 674

11.6.1 Detailed Description . . . 675

11.6.2 Field Documentation . . . 675

11.7 CFE_ES_BlockStats Struct Reference . . . 675

11.7.1 Detailed Description . . . 676

11.7.2 Field Documentation . . . 676

11.8 CFE_ES_CDSRegDumpRec Struct Reference . . . 676

11.8.1 Detailed Description . . . 677

11.8.2 Field Documentation . . . 677

11.9 CFE_ES_DeleteCDSCmd Struct Reference . . . 678

11.9.1 Detailed Description . . . 679

11.9.2 Field Documentation . . . 679

11.10 CFE_ES_DeleteCDSCmd_Payload Struct Reference . . . 679
11.10.1 Detailed Description . . . 680
11.10.2 Field Documentation . . . 680
11.11 CFE_ES_DumpCDSRegistryCmd Struct Reference . . . 680
11.11.1 Detailed Description . . . 680
11.11.2 Field Documentation . . . 681
11.12 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference . . . 681
11.12.1 Detailed Description . . . 681
11.12.2 Field Documentation . . . 682
11.13 CFE_ES_FileNameCmd Struct Reference . . . 682
11.13.1 Detailed Description . . . 682
11.13.2 Field Documentation . . . 682
11.14 CFE_ES_FileNameCmd_Payload Struct Reference . . . 683
11.14.1 Detailed Description . . . 683
11.14.2 Field Documentation . . . 683
11.15 CFE_ES_HousekeepingTlm Struct Reference . . . 684
11.15.1 Detailed Description . . . 684
11.15.2 Field Documentation . . . 684
11.16 CFE_ES_HousekeepingTlm_Payload Struct Reference . . . 684
11.16.1 Detailed Description . . . 686
11.16.2 Field Documentation . . . 687
11.17 CFE_ES_MemPoolStats Struct Reference . . . 697
11.17.1 Detailed Description . . . 698
11.17.2 Field Documentation . . . 698
11.18 CFE_ES_MemStatsTlm Struct Reference . . . 699
11.18.1 Detailed Description . . . 700
11.18.2 Field Documentation . . . 700
11.19 CFE_ES_NoArgsCmd Struct Reference . . . 700

11.19.1 Detailed Description 701
11.19.2 Field Documentation 701
11.20 CFE_ES_OneAppTlm Struct Reference 701
11.20.1 Detailed Description 702
11.20.2 Field Documentation 702
11.21 CFE_ES_OneAppTlm_Payload Struct Reference 702
11.21.1 Detailed Description 702
11.21.2 Field Documentation 703
11.22 CFE_ES_OverWriteSysLogCmd Struct Reference 703
11.22.1 Detailed Description 703
11.22.2 Field Documentation 703
11.23 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference 704
11.23.1 Detailed Description 704
11.23.2 Field Documentation 704
11.24 CFE_ES_PoolAlign Union Reference 705
11.24.1 Detailed Description 705
11.24.2 Field Documentation 705
11.25 CFE_ES_PoolStatsTlm_Payload Struct Reference 706
11.25.1 Detailed Description 706
11.25.2 Field Documentation 706
11.26 CFE_ES_ReloadAppCmd Struct Reference 707
11.26.1 Detailed Description 707
11.26.2 Field Documentation 707
11.27 CFE_ES_RestartCmd Struct Reference 708
11.27.1 Detailed Description 708
11.27.2 Field Documentation 708
11.28 CFE_ES_RestartCmd_Payload Struct Reference 709
11.28.1 Detailed Description 709

11.28.2 Field Documentation 709
11.29 CFE_ES_SendMemPoolStatsCmd Struct Reference 710
11.29.1 Detailed Description 710
11.29.2 Field Documentation 710
11.30 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference 711
11.30.1 Detailed Description 711
11.30.2 Field Documentation 711
11.31 CFE_ES_SetMaxPRCountCmd Struct Reference 712
11.31.1 Detailed Description 712
11.31.2 Field Documentation 712
11.32 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference 713
11.32.1 Detailed Description 713
11.32.2 Field Documentation 713
11.33 CFE_ES_SetPerfFilterMaskCmd Struct Reference 713
11.33.1 Detailed Description 714
11.33.2 Field Documentation 714
11.34 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference 714
11.34.1 Detailed Description 715
11.34.2 Field Documentation 715
11.35 CFE_ES_SetPerfTriggerMaskCmd Struct Reference 715
11.35.1 Detailed Description 716
11.35.2 Field Documentation 716
11.36 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference 716
11.36.1 Detailed Description 717
11.36.2 Field Documentation 717
11.37 CFE_ES_StartApp Struct Reference 717
11.37.1 Detailed Description 718
11.37.2 Field Documentation 718

11.38 CFE_ES_StartAppCmd_Payload Struct Reference . . . 718
11.38.1 Detailed Description . . . 719
11.38.2 Field Documentation . . . 719
11.39 CFE_ES_StartPerfCmd_Payload Struct Reference . . . 720
11.39.1 Detailed Description . . . 721
11.39.2 Field Documentation . . . 721
11.40 CFE_ES_StartPerfDataCmd Struct Reference . . . 721
11.40.1 Detailed Description . . . 722
11.40.2 Field Documentation . . . 722
11.41 CFE_ES_StopPerfCmd_Payload Struct Reference . . . 722
11.41.1 Detailed Description . . . 723
11.41.2 Field Documentation . . . 723
11.42 CFE_ES_StopPerfDataCmd Struct Reference . . . 723
11.42.1 Detailed Description . . . 723
11.42.2 Field Documentation . . . 724
11.43 CFE_ES_TaskInfo Struct Reference . . . 724
11.43.1 Detailed Description . . . 725
11.43.2 Field Documentation . . . 725
11.44 CFE_EVS_AppDataCmd_Payload Struct Reference . . . 727
11.44.1 Detailed Description . . . 727
11.44.2 Field Documentation . . . 727
11.45 CFE_EVS_AppNameBitMaskCmd Struct Reference . . . 727
11.45.1 Detailed Description . . . 728
11.45.2 Field Documentation . . . 728
11.46 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference . . . 728
11.46.1 Detailed Description . . . 729
11.46.2 Field Documentation . . . 729
11.47 CFE_EVS_AppNameCmd Struct Reference . . . 730

11.47.1 Detailed Description 730
11.47.2 Field Documentation 730
11.48 CFE_EVS_AppNameCmd_Payload Struct Reference 731
11.48.1 Detailed Description 731
11.48.2 Field Documentation 731
11.49 CFE_EVS_AppNameEventIDCmd Struct Reference 732
11.49.1 Detailed Description 732
11.49.2 Field Documentation 732
11.50 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference 733
11.50.1 Detailed Description 733
11.50.2 Field Documentation 733
11.51 CFE_EVS_AppNameEventIDMaskCmd Struct Reference 734
11.51.1 Detailed Description 734
11.51.2 Field Documentation 734
11.52 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference 735
11.52.1 Detailed Description 735
11.52.2 Field Documentation 735
11.53 CFE_EVS_AppTlmData Struct Reference 736
11.53.1 Detailed Description 736
11.53.2 Field Documentation 736
11.54 CFE_EVS_BinFilter Struct Reference 737
11.54.1 Detailed Description 738
11.54.2 Field Documentation 738
11.55 CFE_EVS_BitMaskCmd Struct Reference 738
11.55.1 Detailed Description 739
11.55.2 Field Documentation 739
11.56 CFE_EVS_BitMaskCmd_Payload Struct Reference 739
11.56.1 Detailed Description 740

11.56.2 Field Documentation 740
11.57 CFE_EVS_HousekeepingTlm Struct Reference 740
11.57.1 Detailed Description 741
11.57.2 Field Documentation 741
11.58 CFE_EVS_HousekeepingTlm_Payload Struct Reference 741
11.58.1 Detailed Description 742
11.58.2 Field Documentation 742
11.59 CFE_EVS_LogFileCmd_Payload Struct Reference 746
11.59.1 Detailed Description 747
11.59.2 Field Documentation 747
11.60 CFE_EVS_LongEventTlm Struct Reference 747
11.60.1 Detailed Description 747
11.60.2 Field Documentation 748
11.61 CFE_EVS_LongEventTlm_Payload Struct Reference 748
11.61.1 Detailed Description 748
11.61.2 Field Documentation 749
11.62 CFE_EVS_NoArgsCmd Struct Reference 750
11.62.1 Detailed Description 750
11.62.2 Field Documentation 750
11.63 CFE_EVS_PacketID Struct Reference 750
11.63.1 Detailed Description 751
11.63.2 Field Documentation 751
11.64 CFE_EVS_SetEventFormatCode_Payload Struct Reference 752
11.64.1 Detailed Description 752
11.64.2 Field Documentation 753
11.65 CFE_EVS_SetEventFormatModeCmd Struct Reference 753
11.65.1 Detailed Description 753
11.65.2 Field Documentation 754

11.66 CFE_EVS_SetLogMode_Payload Struct Reference . . . 754
11.66.1 Detailed Description . . . 754
11.66.2 Field Documentation . . . 755
11.67 CFE_EVS_SetLogModeCmd Struct Reference . . . 755
11.67.1 Detailed Description . . . 755
11.67.2 Field Documentation . . . 756
11.68 CFE_EVS_ShortEventTlm Struct Reference . . . 756
11.68.1 Detailed Description . . . 756
11.68.2 Field Documentation . . . 756
11.69 CFE_EVS_ShortEventTlm_Payload Struct Reference . . . 757
11.69.1 Detailed Description . . . 757
11.69.2 Field Documentation . . . 757
11.70 CFE_EVS_WriteAppDataFileCmd Struct Reference . . . 758
11.70.1 Detailed Description . . . 758
11.70.2 Field Documentation . . . 758
11.71 CFE_EVS_WriteLogDataFileCmd Struct Reference . . . 759
11.71.1 Detailed Description . . . 759
11.71.2 Field Documentation . . . 759
11.72 CFE_FS_FileWriteMetaData Struct Reference . . . 760
11.72.1 Detailed Description . . . 760
11.72.2 Field Documentation . . . 760
11.73 CFE_FS_Header Struct Reference . . . 762
11.73.1 Detailed Description . . . 762
11.73.2 Field Documentation . . . 762
11.74 CFE_SB_AllSubscriptionsTlm Struct Reference . . . 764
11.74.1 Detailed Description . . . 765
11.74.2 Field Documentation . . . 765
11.75 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference . . . 765

11.75.1 Detailed Description 766
11.75.2 Field Documentation 766
11.76 CFE_SB_HousekeepingTlm Struct Reference 767
11.76.1 Detailed Description 767
11.76.2 Field Documentation 767
11.77 CFE_SB_HousekeepingTlm_Payload Struct Reference 768
11.77.1 Detailed Description 769
11.77.2 Field Documentation 769
11.78 CFE_SB_Msg Union Reference 773
11.78.1 Detailed Description 773
11.78.2 Field Documentation 774
11.79 CFE_SB_MsgId_t Struct Reference 774
11.79.1 Detailed Description 775
11.79.2 Field Documentation 775
11.80 CFE_SB_MsgMapFileEntry Struct Reference 775
11.80.1 Detailed Description 776
11.80.2 Field Documentation 776
11.81 CFE_SB_PipeDepthStats Struct Reference 776
11.81.1 Detailed Description 777
11.81.2 Field Documentation 777
11.82 CFE_SB_PipeInfoEntry Struct Reference 778
11.82.1 Detailed Description 779
11.82.2 Field Documentation 779
11.83 CFE_SB_Qos_t Struct Reference 781
11.83.1 Detailed Description 782
11.83.2 Field Documentation 782
11.84 CFE_SB_RouteCmd Struct Reference 782
11.84.1 Detailed Description 783

11.84.2 Field Documentation 783
11.85 CFE_SB_RouteCmd_Payload Struct Reference 783
11.85.1 Detailed Description 784
11.85.2 Field Documentation 784
11.86 CFE_SB_RoutingFileEntry Struct Reference 785
11.86.1 Detailed Description 785
11.86.2 Field Documentation 785
11.87 CFE_SB_SingleSubscriptionTlm Struct Reference 787
11.87.1 Detailed Description 787
11.87.2 Field Documentation 787
11.88 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference 787
11.88.1 Detailed Description 788
11.88.2 Field Documentation 788
11.89 CFE_SB_StatsTlm Struct Reference 789
11.89.1 Detailed Description 789
11.89.2 Field Documentation 789
11.90 CFE_SB_StatsTlm_Payload Struct Reference 790
11.90.1 Detailed Description 791
11.90.2 Field Documentation 791
11.91 CFE_SB_SubEntries Struct Reference 795
11.91.1 Detailed Description 796
11.91.2 Field Documentation 796
11.92 CFE_SB_WriteFileInfoCmd Struct Reference 797
11.92.1 Detailed Description 797
11.92.2 Field Documentation 797
11.93 CFE_SB_WriteFileInfoCmd_Payload Struct Reference 798
11.93.1 Detailed Description 798
11.93.2 Field Documentation 798

11.94 CFE_TBL_AbortLoadCmd Struct Reference . . . 798
11.94.1 Detailed Description . . . 799
11.94.2 Field Documentation . . . 799
11.95 CFE_TBL_AbortLoadCmd_Payload Struct Reference . . . 799
11.95.1 Detailed Description . . . 800
11.95.2 Field Documentation . . . 800
11.96 CFE_TBL_ActivateCmd Struct Reference . . . 800
11.96.1 Detailed Description . . . 801
11.96.2 Field Documentation . . . 801
11.97 CFE_TBL_ActivateCmd_Payload Struct Reference . . . 801
11.97.1 Detailed Description . . . 802
11.97.2 Field Documentation . . . 802
11.98 CFE_TBL_DelCDSCmd_Payload Struct Reference . . . 802
11.98.1 Detailed Description . . . 802
11.98.2 Field Documentation . . . 803
11.99 CFE_TBL_DeleteCDSCmd Struct Reference . . . 803
11.99.1 Detailed Description . . . 803
11.99.2 Field Documentation . . . 803
11.100 CFE_TBL_DumpCmd Struct Reference . . . 804
11.100.1 Detailed Description . . . 804
11.100.2 Field Documentation . . . 804
11.101 CFE_TBL_DumpCmd_Payload Struct Reference . . . 805
11.101.1 Detailed Description . . . 805
11.101.2 Field Documentation . . . 806
11.102 CFE_TBL_DumpRegistryCmd Struct Reference . . . 806
11.102.1 Detailed Description . . . 807
11.102.2 Field Documentation . . . 807
11.103 CFE_TBL_DumpRegistryCmd_Payload Struct Reference . . . 807

11.103.1 Detailed Description . . . 808
11.103.2 Field Documentation . . . 808
11.104 CFE_TBL_File_Hdr Struct Reference . . . 808
11.104.1 Detailed Description . . . 809
11.104.2 Field Documentation . . . 809
11.105 CFE_TBL_FileDef Struct Reference . . . 810
11.105.1 Detailed Description . . . 810
11.105.2 Field Documentation . . . 810
11.106 CFE_TBL_HousekeepingTlm Struct Reference . . . 811
11.106.1 Detailed Description . . . 811
11.106.2 Field Documentation . . . 812
11.107 CFE_TBL_HousekeepingTlm_Payload Struct Reference . . . 812
11.107.1 Detailed Description . . . 813
11.107.2 Field Documentation . . . 813
11.108 CFE_TBL_Info Struct Reference . . . 818
11.108.1 Detailed Description . . . 819
11.108.2 Field Documentation . . . 819
11.109 CFE_TBL_LoadCmd Struct Reference . . . 822
11.109.1 Detailed Description . . . 822
11.109.2 Field Documentation . . . 822
11.110 CFE_TBL_LoadCmd_Payload Struct Reference . . . 823
11.110.1 Detailed Description . . . 823
11.110.2 Field Documentation . . . 823
11.111 CFE_TBL_NoArgsCmd Struct Reference . . . 824
11.111.1 Detailed Description . . . 824
11.111.2 Field Documentation . . . 824
11.112 CFE_TBL_NotifyCmd Struct Reference . . . 824
11.112.1 Detailed Description . . . 825

11.112.2 Field Documentation . . . 825
11.113 CFE_TBL_NotifyCmd_Payload Struct Reference . . . 825
11.113.1 Detailed Description . . . 826
11.113.2 Field Documentation . . . 826
11.114 CFE_TBL_SendRegistryCmd Struct Reference . . . 826
11.114.1 Detailed Description . . . 827
11.114.2 Field Documentation . . . 827
11.115 CFE_TBL_SendRegistryCmd_Payload Struct Reference . . . 827
11.115.1 Detailed Description . . . 828
11.115.2 Field Documentation . . . 828
11.116 CFE_TBL_TableRegistryTlm Struct Reference . . . 828
11.116.1 Detailed Description . . . 828
11.116.2 Field Documentation . . . 829
11.117 CFE_TBL_TblRegPacket_Payload Struct Reference . . . 829
11.117.1 Detailed Description . . . 830
11.117.2 Field Documentation . . . 830
11.118 CFE_TBL_ValidateCmd Struct Reference . . . 835
11.118.1 Detailed Description . . . 835
11.118.2 Field Documentation . . . 835
11.119 CFE_TBL_ValidateCmd_Payload Struct Reference . . . 836
11.119.1 Detailed Description . . . 836
11.119.2 Field Documentation . . . 836
11.120 CFE_TIME_DiagnosticTlm Struct Reference . . . 837
11.120.1 Detailed Description . . . 837
11.120.2 Field Documentation . . . 837
11.121 CFE_TIME_DiagnosticTlm_Payload Struct Reference . . . 837
11.121.1 Detailed Description . . . 839
11.121.2 Field Documentation . . . 840

11.122 CFE_TIME_HousekeepingTlm Struct Reference 850
11.122.1 Detailed Description 850
11.122.2 Field Documentation 851
11.123 CFE_TIME_HousekeepingTlm_Payload Struct Reference 851
11.123.1 Detailed Description 852
11.123.2 Field Documentation 852
11.124 CFE_TIME_LeapsCmd_Payload Struct Reference 855
11.124.1 Detailed Description 856
11.124.2 Field Documentation 856
11.125 CFE_TIME_NoArgsCmd Struct Reference 856
11.125.1 Detailed Description 856
11.125.2 Field Documentation 857
11.126 CFE_TIME_OneHzAdjustmentCmd Struct Reference 857
11.126.1 Detailed Description 857
11.126.2 Field Documentation 857
11.127 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference 858
11.127.1 Detailed Description 858
11.127.2 Field Documentation 858
11.128 CFE_TIME_SetLeapSecondsCmd Struct Reference 859
11.128.1 Detailed Description 859
11.128.2 Field Documentation 859
11.129 CFE_TIME_SetSignalCmd Struct Reference 860
11.129.1 Detailed Description 860
11.129.2 Field Documentation 860
11.130 CFE_TIME_SetSourceCmd Struct Reference 861
11.130.1 Detailed Description 861
11.130.2 Field Documentation 861
11.131 CFE_TIME_SetStateCmd Struct Reference 862

11.131.1 Detailed Description 862
11.131.2 Field Documentation 862
11.132 CFE_TIME_SignalCmd_Payload Struct Reference 863
11.132.1 Detailed Description 863
11.132.2 Field Documentation 863
11.133 CFE_TIME_SourceCmd_Payload Struct Reference 864
11.133.1 Detailed Description 864
11.133.2 Field Documentation 864
11.134 CFE_TIME_StateCmd_Payload Struct Reference 864
11.134.1 Detailed Description 865
11.134.2 Field Documentation 865
11.135 CFE_TIME_SysTime Struct Reference 865
11.135.1 Detailed Description 865
11.135.2 Field Documentation 866
11.136 CFE_TIME_TimeCmd Struct Reference 866
11.136.1 Detailed Description 866
11.136.2 Field Documentation 867
11.137 CFE_TIME_TimeCmd_Payload Struct Reference 867
11.137.1 Detailed Description 867
11.137.2 Field Documentation 867
11.138 CFE_TIME_ToneDataCmd Struct Reference 868
11.138.1 Detailed Description 868
11.138.2 Field Documentation 868
11.139 CFE_TIME_ToneDataCmd_Payload Struct Reference 869
11.139.1 Detailed Description 869
11.139.2 Field Documentation 870
11.140 FM_ChildQueueEntry_t Struct Reference 871
11.140.1 Detailed Description 871

11.140.2 Field Documentation . . . 871
11.141 FM_ConcatCmd_t Struct Reference . . . 874
11.141.1 Detailed Description . . . 875
11.141.2 Field Documentation . . . 875
11.142 FM_CopyFileCmd_t Struct Reference . . . 876
11.142.1 Detailed Description . . . 876
11.142.2 Field Documentation . . . 877
11.143 FM_CreateDirCmd_t Struct Reference . . . 878
11.143.1 Detailed Description . . . 878
11.143.2 Field Documentation . . . 878
11.144 FM_DecompressCmd_t Struct Reference . . . 879
11.144.1 Detailed Description . . . 879
11.144.2 Field Documentation . . . 879
11.145 FM_DeleteAllCmd_t Struct Reference . . . 880
11.145.1 Detailed Description . . . 880
11.145.2 Field Documentation . . . 880
11.146 FM_DeleteDirCmd_t Struct Reference . . . 881
11.146.1 Detailed Description . . . 881
11.146.2 Field Documentation . . . 881
11.147 FM_DeleteFileCmd_t Struct Reference . . . 882
11.147.1 Detailed Description . . . 882
11.147.2 Field Documentation . . . 882
11.148 FM_DirListEntry_t Struct Reference . . . 883
11.148.1 Detailed Description . . . 883
11.148.2 Field Documentation . . . 883
11.149 FM_DirListFileStats_t Struct Reference . . . 885
11.149.1 Detailed Description . . . 885
11.149.2 Field Documentation . . . 885

11.150 FM_DirListPkt_t Struct Reference . . . 886
11.150.1 Detailed Description . . . 886
11.150.2 Field Documentation . . . 886
11.151 FM_FileInfoPkt_t Struct Reference . . . 888
11.151.1 Detailed Description . . . 889
11.151.2 Field Documentation . . . 889
11.152 FM_FreeSpacePkt_t Struct Reference . . . 891
11.152.1 Detailed Description . . . 891
11.152.2 Field Documentation . . . 891
11.153 FM_FreeSpacePktEntry_t Struct Reference . . . 892
11.153.1 Detailed Description . . . 892
11.153.2 Field Documentation . . . 893
11.154 FM_FreeSpaceTable_t Struct Reference . . . 893
11.154.1 Detailed Description . . . 893
11.154.2 Field Documentation . . . 894
11.155 FM_GetDirFileCmd_t Struct Reference . . . 894
11.155.1 Detailed Description . . . 894
11.155.2 Field Documentation . . . 895
11.156 FM_GetDirPktCmd_t Struct Reference . . . 896
11.156.1 Detailed Description . . . 896
11.156.2 Field Documentation . . . 896
11.157 FM_GetFileInfoCmd_t Struct Reference . . . 898
11.157.1 Detailed Description . . . 898
11.157.2 Field Documentation . . . 898
11.158 FM_GetFreeSpaceCmd_t Struct Reference . . . 899
11.158.1 Detailed Description . . . 899
11.158.2 Field Documentation . . . 900
11.159 FM_GetOpenFilesCmd_t Struct Reference . . . 900

11.159.1 Detailed Description 900
11.159.2 Field Documentation 900
11.160 FM_GlobalData_t Struct Reference 901
11.160.1 Detailed Description 902
11.160.2 Field Documentation 902
11.161 FM_HousekeepingCmd_t Struct Reference 910
11.161.1 Detailed Description 910
11.161.2 Field Documentation 910
11.162 FM_HousekeepingPkt_t Struct Reference 911
11.162.1 Detailed Description 911
11.162.2 Field Documentation 911
11.163 FM_MoveFileCmd_t Struct Reference 914
11.163.1 Detailed Description 915
11.163.2 Field Documentation 915
11.164 FM_NoopCmd_t Struct Reference 916
11.164.1 Detailed Description 916
11.164.2 Field Documentation 916
11.165 FM_OpenFilesEntry_t Struct Reference 917
11.165.1 Detailed Description 917
11.165.2 Field Documentation 917
11.166 FM_OpenFilesPkt_t Struct Reference 918
11.166.1 Detailed Description 918
11.166.2 Field Documentation 918
11.167 FM_RenameFileCmd_t Struct Reference 919
11.167.1 Detailed Description 919
11.167.2 Field Documentation 919
11.168 FM_ResetCmd_t Struct Reference 920
11.168.1 Detailed Description 921

11.168.2 Field Documentation . . . 921
11.169 FM_SetPermCmd_t Struct Reference . . . 921
11.169.1 Detailed Description . . . 921
11.169.2 Field Documentation . . . 922
11.170 FM_SetTableStateCmd_t Struct Reference . . . 922
11.170.1 Detailed Description . . . 923
11.170.2 Field Documentation . . . 923
11.171 FM_TableEntry_t Struct Reference . . . 924
11.171.1 Detailed Description . . . 924
11.171.2 Field Documentation . . . 924
11.172 OS_bin_sem_prop_t Struct Reference . . . 925
11.172.1 Detailed Description . . . 925
11.172.2 Field Documentation . . . 925
11.173 OS_count_sem_prop_t Struct Reference . . . 926
11.173.1 Detailed Description . . . 926
11.173.2 Field Documentation . . . 926
11.174 os_dirent_t Struct Reference . . . 927
11.174.1 Detailed Description . . . 927
11.174.2 Field Documentation . . . 927
11.175 OS_FdSet Struct Reference . . . 928
11.175.1 Detailed Description . . . 928
11.175.2 Field Documentation . . . 928
11.176 OS_file_prop_t Struct Reference . . . 928
11.176.1 Detailed Description . . . 929
11.176.2 Field Documentation . . . 929
11.177 os_fsinfo_t Struct Reference . . . 930
11.177.1 Detailed Description . . . 930
11.177.2 Field Documentation . . . 930

11.178 os_fstat_t Struct Reference . . . 931
11.178.1 Detailed Description . . . 931
11.178.2 Field Documentation . . . 932
11.179 OS_heap_prop_t Struct Reference . . . 932
11.179.1 Detailed Description . . . 933
11.179.2 Field Documentation . . . 933
11.180 OS_module_address_t Struct Reference . . . 933
11.180.1 Detailed Description . . . 934
11.180.2 Field Documentation . . . 934
11.181 OS_module_prop_t Struct Reference . . . 935
11.181.1 Detailed Description . . . 936
11.181.2 Field Documentation . . . 936
11.182 OS_mut_sem_prop_t Struct Reference . . . 937
11.182.1 Detailed Description . . . 937
11.182.2 Field Documentation . . . 937
11.183 OS_queue_prop_t Struct Reference . . . 938
11.183.1 Detailed Description . . . 938
11.183.2 Field Documentation . . . 938
11.184 OS_SockAddr_t Struct Reference . . . 938
11.184.1 Detailed Description . . . 939
11.184.2 Field Documentation . . . 939
11.185 OS_SockAddrData_t Union Reference . . . 939
11.185.1 Detailed Description . . . 940
11.185.2 Field Documentation . . . 940
11.186 OS_socket_prop_t Struct Reference . . . 941
11.186.1 Detailed Description . . . 941
11.186.2 Field Documentation . . . 941
11.187 OS_static_symbol_record_t Struct Reference . . . 942

11.187.1 Detailed Description 942
11.187.2 Field Documentation 942
11.188 OS_statvfs_t Struct Reference 943
11.188.1 Detailed Description 943
11.188.2 Field Documentation 943
11.189 OS_task_prop_t Struct Reference 944
11.189.1 Detailed Description 944
11.189.2 Field Documentation 944
11.190 OS_time_t Struct Reference 945
11.190.1 Detailed Description 946
11.190.2 Field Documentation 946
11.191 OS_timebase_prop_t Struct Reference 946
11.191.1 Detailed Description 947
11.191.2 Field Documentation 947
11.192 OS_timer_prop_t Struct Reference 948
11.192.1 Detailed Description 948
11.192.2 Field Documentation 948

12 File Documentation **949**

12.1 apps/fm/docs/dox_src/cfs_fm.dox File Reference 949
12.2 apps/fm/fsw/mission_inc/fm_perfids.h File Reference 949
12.2.1 Detailed Description 949
12.3 apps/fm/fsw/platform_inc/fm_msgids.h File Reference 949
12.3.1 Detailed Description 950
12.4 apps/fm/fsw/platform_inc/fm_platform_cfg.h File Reference 950
12.4.1 Detailed Description 951
12.5 apps/fm/fsw/src/fm_app.c File Reference 951
12.5.1 Detailed Description 952

12.5.2 Function Documentation . . . 952
12.5.3 Variable Documentation . . . 962
12.6 apps/fm/fsw/src/fm_app.h File Reference . . . 962
12.6.1 Detailed Description . . . 963
12.6.2 Macro Definition Documentation . . . 963
12.6.3 Function Documentation . . . 963
12.7 apps/fm/fsw/src/fm_child.c File Reference . . . 972
12.7.1 Detailed Description . . . 973
12.7.2 Macro Definition Documentation . . . 973
12.7.3 Function Documentation . . . 974
12.8 apps/fm/fsw/src/fm_child.h File Reference . . . 1000
12.8.1 Detailed Description . . . 1001
12.8.2 Function Documentation . . . 1001
12.9 apps/fm/fsw/src/fm_cmd_utils.c File Reference . . . 1028
12.9.1 Detailed Description . . . 1029
12.9.2 Function Documentation . . . 1029
12.9.3 Variable Documentation . . . 1048
12.10 apps/fm/fsw/src/fm_cmd_utils.h File Reference . . . 1048
12.10.1 Detailed Description . . . 1049
12.10.2 Function Documentation . . . 1049
12.11 apps/fm/fsw/src/fm_cmds.c File Reference . . . 1066
12.11.1 Detailed Description . . . 1067
12.11.2 Function Documentation . . . 1067
12.12 apps/fm/fsw/src/fm_cmds.h File Reference . . . 1090
12.12.1 Detailed Description . . . 1091
12.12.2 Function Documentation . . . 1091
12.13 apps/fm/fsw/src/fm_defs.h File Reference . . . 1114
12.13.1 Detailed Description . . . 1114

12.13.2 Macro Definition Documentation . . . 1115
12.14 apps/fm/fsw/src/fm_events.h File Reference . . . 1117
12.14.1 Detailed Description . . . 1129
12.15 apps/fm/fsw/src/fm_msg.h File Reference . . . 1129
12.15.1 Detailed Description . . . 1131
12.15.2 Variable Documentation . . . 1131
12.16 apps/fm/fsw/src/fm_msgdefs.h File Reference . . . 1132
12.16.1 Detailed Description . . . 1132
12.17 apps/fm/fsw/src/fm_tbl.c File Reference . . . 1133
12.17.1 Detailed Description . . . 1133
12.17.2 Function Documentation . . . 1133
12.18 apps/fm/fsw/src/fm_tbl.h File Reference . . . 1137
12.18.1 Detailed Description . . . 1138
12.18.2 Function Documentation . . . 1138
12.19 apps/fm/fsw/src/fm_verify.h File Reference . . . 1142
12.19.1 Detailed Description . . . 1143
12.20 apps/fm/fsw/src/fm_version.h File Reference . . . 1143
12.20.1 Detailed Description . . . 1143
12.21 apps/fm/fsw/tables/fm_freespace.c File Reference . . . 1143
12.21.1 Detailed Description . . . 1143
12.21.2 Function Documentation . . . 1144
12.21.3 Variable Documentation . . . 1144
12.22 build/docs/osconfig-example.h File Reference . . . 1144
12.22.1 Macro Definition Documentation . . . 1145
12.23 cfe/cmake/sample_defs/cpu1_msgids.h File Reference . . . 1153
12.23.1 Detailed Description . . . 1154
12.23.2 Macro Definition Documentation . . . 1155
12.24 cfe/cmake/sample_defs/cpu1_platform_cfg.h File Reference . . . 1161

12.24.1 Detailed Description 1164
12.24.2 Macro Definition Documentation 1164
12.25 cfe/cmake/sample_defs/sample_mission_cfg.h File Reference 1218
12.25.1 Detailed Description 1220
12.25.2 Macro Definition Documentation 1220
12.26 cfe/cmake/sample_defs/sample_perfids.h File Reference 1239
12.26.1 Detailed Description 1239
12.26.2 Macro Definition Documentation 1239
12.27 cfe/docs/src/cfe_api.dox File Reference 1242
12.28 cfe/docs/src/cfe_es.dox File Reference 1242
12.29 cfe/docs/src/cfe_evs.dox File Reference 1242
12.30 cfe/docs/src/cfe_frontpage.dox File Reference 1242
12.31 cfe/docs/src/cfe_glossary.dox File Reference 1242
12.32 cfe/docs/src/cfe_sb.dox File Reference 1242
12.33 cfe/docs/src/cfe_tbl.dox File Reference 1242
12.34 cfe/docs/src/cfe_time.dox File Reference 1242
12.35 cfe/docs/src/cfe_xref.dox File Reference 1242
12.36 cfe/docs/src/cfs_versions.dox File Reference 1243
12.37 cfe/modules/core_api/fsw/inc/cfe.h File Reference 1243
12.37.1 Detailed Description 1243
12.38 cfe/modules/core_api/fsw/inc/cfe_config.h File Reference 1243
12.38.1 Detailed Description 1244
12.38.2 Function Documentation 1244
12.39 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference 1247
12.39.1 Detailed Description 1247
12.39.2 Macro Definition Documentation 1247
12.39.3 Typedef Documentation 1248
12.40 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference 1248

12.40.1 Detailed Description 1248
12.40.2 Macro Definition Documentation 1249
12.41 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference 1249
12.41.1 Detailed Description 1255
12.41.2 Macro Definition Documentation 1255
12.41.3 Typedef Documentation 1258
12.42 cfe/modules/core_api/fsw/inc/cfe_es.h File Reference 1258
12.42.1 Detailed Description 1261
12.42.2 Macro Definition Documentation 1261
12.43 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference 1262
12.43.1 Detailed Description 1264
12.43.2 Macro Definition Documentation 1264
12.43.3 Typedef Documentation 1267
12.44 cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h File Reference 1269
12.44.1 Detailed Description 1271
12.44.2 Macro Definition Documentation 1271
12.44.3 Typedef Documentation 1272
12.44.4 Enumeration Type Documentation 1278
12.45 cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference 1281
12.45.1 Detailed Description 1281
12.45.2 Macro Definition Documentation 1281
12.46 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference 1283
12.46.1 Detailed Description 1283
12.46.2 Macro Definition Documentation 1284
12.46.3 Typedef Documentation 1286
12.47 cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h File Reference 1286
12.47.1 Detailed Description 1287
12.47.2 Typedef Documentation 1287

12.47.3 Enumeration Type Documentation . . . 1288
12.48 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference . . . 1291
12.48.1 Detailed Description . . . 1292
12.49 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference . . . 1292
12.49.1 Detailed Description . . . 1292
12.49.2 Typedef Documentation . . . 1293
12.49.3 Enumeration Type Documentation . . . 1294
12.50 cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h File Reference . . . 1295
12.50.1 Detailed Description . . . 1296
12.50.2 Macro Definition Documentation . . . 1296
12.50.3 Typedef Documentation . . . 1296
12.50.4 Enumeration Type Documentation . . . 1297
12.51 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference . . . 1298
12.51.1 Detailed Description . . . 1300
12.52 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference . . . 1300
12.52.1 Detailed Description . . . 1302
12.52.2 Macro Definition Documentation . . . 1302
12.52.3 Typedef Documentation . . . 1302
12.52.4 Enumeration Type Documentation . . . 1305
12.53 cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference . . . 1308
12.53.1 Detailed Description . . . 1309
12.53.2 Macro Definition Documentation . . . 1309
12.53.3 Function Documentation . . . 1310
12.54 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference . . . 1315
12.54.1 Detailed Description . . . 1316
12.54.2 Macro Definition Documentation . . . 1316
12.55 cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference . . . 1317
12.55.1 Detailed Description . . . 1318

12.55.2 Macro Definition Documentation 1318
12.56 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference 1319
12.56.1 Detailed Description 1320
12.56.2 Macro Definition Documentation 1321
12.56.3 Typedef Documentation 1324
12.57 cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h File Reference 1324
12.57.1 Detailed Description 1325
12.57.2 Macro Definition Documentation 1325
12.57.3 Typedef Documentation 1326
12.57.4 Enumeration Type Documentation 1327
12.58 cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference 1328
12.58.1 Detailed Description 1329
12.59 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference 1329
12.59.1 Detailed Description 1330
12.59.2 Macro Definition Documentation 1331
12.59.3 Typedef Documentation 1331
12.59.4 Enumeration Type Documentation 1332
12.60 cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h File Reference 1332
12.60.1 Detailed Description 1333
12.60.2 Typedef Documentation 1333
12.60.3 Enumeration Type Documentation 1333
12.61 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h File Reference 1334
12.61.1 Detailed Description 1334
12.61.2 Macro Definition Documentation 1335
12.61.3 Typedef Documentation 1335
12.62 cfe/modules/core_api/fsw/inc/cfe_time.h File Reference 1336
12.62.1 Detailed Description 1337
12.62.2 Macro Definition Documentation 1337

12.63 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference . . . 1338
12.63.1 Detailed Description . . . 1338
12.63.2 Macro Definition Documentation . . . 1338
12.63.3 Typedef Documentation . . . 1339
12.63.4 Enumeration Type Documentation . . . 1339
12.64 cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h File Reference . . . 1340
12.64.1 Detailed Description . . . 1341
12.64.2 Typedef Documentation . . . 1341
12.64.3 Enumeration Type Documentation . . . 1343
12.65 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference . . . 1346
12.65.1 Detailed Description . . . 1346
12.65.2 Macro Definition Documentation . . . 1347
12.66 cfe/modules/es/fsw/inc/cfe_es_events.h File Reference . . . 1349
12.66.1 Detailed Description . . . 1352
12.66.2 Macro Definition Documentation . . . 1352
12.67 cfe/modules/es/fsw/inc/cfe_es_msg.h File Reference . . . 1382
12.67.1 Detailed Description . . . 1386
12.67.2 Macro Definition Documentation . . . 1386
12.67.3 Typedef Documentation . . . 1410
12.68 cfe/modules/evs/fsw/inc/cfe_evs_events.h File Reference . . . 1418
12.68.1 Detailed Description . . . 1419
12.68.2 Macro Definition Documentation . . . 1419
12.69 cfe/modules/evs/fsw/inc/cfe_evs_msg.h File Reference . . . 1432
12.69.1 Detailed Description . . . 1436
12.69.2 Macro Definition Documentation . . . 1436
12.69.3 Typedef Documentation . . . 1457
12.70 cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference . . . 1464
12.70.1 Detailed Description . . . 1465

12.70.2 Typedef Documentation 1465
12.71 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference 1465
12.71.1 Detailed Description 1466
12.72 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference 1466
12.72.1 Detailed Description 1467
12.72.2 Macro Definition Documentation 1467
12.73 cfe/modules/sb/fsw/inc/cfe_sb_events.h File Reference 1468
12.73.1 Detailed Description 1470
12.73.2 Macro Definition Documentation 1470
12.74 cfe/modules/sb/fsw/inc/cfe_sb_msg.h File Reference 1492
12.74.1 Detailed Description 1494
12.74.2 Macro Definition Documentation 1494
12.74.3 Typedef Documentation 1505
12.75 cfe/modules/tbl/fsw/inc/cfe_tbl_events.h File Reference 1510
12.75.1 Detailed Description 1512
12.75.2 Macro Definition Documentation 1512
12.76 cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h File Reference 1536
12.76.1 Detailed Description 1538
12.76.2 Macro Definition Documentation 1538
12.76.3 Typedef Documentation 1549
12.77 cfe/modules/time/fsw/inc/cfe_time_events.h File Reference 1553
12.77.1 Detailed Description 1554
12.77.2 Macro Definition Documentation 1555
12.78 cfe/modules/time/fsw/inc/cfe_time_msg.h File Reference 1567
12.78.1 Detailed Description 1570
12.78.2 Macro Definition Documentation 1570
12.78.3 Typedef Documentation 1587
12.79 osal/docs/src/osal_frontpage.dox File Reference 1593

12.80 osal/docs/src/osal_fs.dox File Reference . . . 1593
12.81 osal/docs/src/osal_timer.dox File Reference . . . 1593
12.82 osal/src/os/inc/common_types.h File Reference . . . 1593
12.82.1 Detailed Description . . . 1595
12.82.2 Macro Definition Documentation . . . 1595
12.82.3 Typedef Documentation . . . 1596
12.82.4 Function Documentation . . . 1600
12.83 osal/src/os/inc/osapi-binsem.h File Reference . . . 1601
12.83.1 Detailed Description . . . 1602
12.84 osal/src/os/inc/osapi-bsp.h File Reference . . . 1602
12.84.1 Detailed Description . . . 1603
12.85 osal/src/os/inc/osapi-clock.h File Reference . . . 1603
12.85.1 Detailed Description . . . 1604
12.85.2 Enumeration Type Documentation . . . 1604
12.86 osal/src/os/inc/osapi-common.h File Reference . . . 1605
12.86.1 Detailed Description . . . 1605
12.86.2 Typedef Documentation . . . 1606
12.86.3 Enumeration Type Documentation . . . 1607
12.87 osal/src/os/inc/osapi-constants.h File Reference . . . 1608
12.87.1 Detailed Description . . . 1608
12.87.2 Macro Definition Documentation . . . 1608
12.88 osal/src/os/inc/osapi-countsem.h File Reference . . . 1609
12.88.1 Detailed Description . . . 1610
12.89 osal/src/os/inc/osapi-dir.h File Reference . . . 1610
12.89.1 Detailed Description . . . 1611
12.89.2 Macro Definition Documentation . . . 1611
12.90 osal/src/os/inc/osapi-error.h File Reference . . . 1611
12.90.1 Detailed Description . . . 1614

12.90.2 Macro Definition Documentation 1614
12.90.3 Typedef Documentation 1614
12.91 osal/src/os/inc/osapi-file.h File Reference 1615
12.91.1 Detailed Description 1616
12.91.2 Macro Definition Documentation 1616
12.91.3 Enumeration Type Documentation 1618
12.92 osal/src/os/inc/osapi-filesys.h File Reference 1619
12.92.1 Detailed Description 1620
12.92.2 Macro Definition Documentation 1620
12.93 osal/src/os/inc/osapi-heap.h File Reference 1621
12.93.1 Detailed Description 1621
12.94 osal/src/os/inc/osapi-idmap.h File Reference 1621
12.94.1 Detailed Description 1623
12.94.2 Macro Definition Documentation 1623
12.95 osal/src/os/inc/osapi-macros.h File Reference 1623
12.95.1 Detailed Description 1624
12.95.2 Macro Definition Documentation 1624
12.96 osal/src/os/inc/osapi-module.h File Reference 1626
12.96.1 Detailed Description 1626
12.96.2 Macro Definition Documentation 1626
12.97 osal/src/os/inc/osapi-mutex.h File Reference 1627
12.97.1 Detailed Description 1628
12.98 osal/src/os/inc/osapi-network.h File Reference 1628
12.98.1 Detailed Description 1628
12.99 osal/src/os/inc/osapi-printf.h File Reference 1629
12.99.1 Detailed Description 1629
12.100 osal/src/os/inc/osapi-queue.h File Reference 1629
12.100.1 Detailed Description 1630

12.101 osal/src/os/inc/osapi-select.h File Reference . . . 1630
12.101.1 Detailed Description . . . 1630
12.101.2 Enumeration Type Documentation . . . 1630
12.102 osal/src/os/inc/osapi-shell.h File Reference . . . 1631
12.102.1 Detailed Description . . . 1631
12.103 osal/src/os/inc/osapi-sockets.h File Reference . . . 1631
12.103.1 Detailed Description . . . 1633
12.103.2 Macro Definition Documentation . . . 1633
12.103.3 Enumeration Type Documentation . . . 1633
12.104 osal/src/os/inc/osapi-task.h File Reference . . . 1634
12.104.1 Detailed Description . . . 1636
12.104.2 Macro Definition Documentation . . . 1636
12.104.3 Typedef Documentation . . . 1637
12.104.4 Function Documentation . . . 1637
12.105 osal/src/os/inc/osapi-timebase.h File Reference . . . 1638
12.105.1 Detailed Description . . . 1638
12.105.2 Typedef Documentation . . . 1639
12.106 osal/src/os/inc/osapi-timer.h File Reference . . . 1639
12.106.1 Detailed Description . . . 1640
12.106.2 Typedef Documentation . . . 1640
12.107 osal/src/os/inc/osapi-version.h File Reference . . . 1640
12.107.1 Detailed Description . . . 1641
12.107.2 Macro Definition Documentation . . . 1641
12.107.3 Function Documentation . . . 1644
12.108 osal/src/os/inc/osapi.h File Reference . . . 1645
12.108.1 Detailed Description . . . 1646
12.109 psp/fsw/inc/cfe_psp.h File Reference . . . 1646
12.109.1 Macro Definition Documentation . . . 1649
12.109.2 Function Documentation . . . 1658

Index 1671

1 CFS File Manager (FM) Documentation

- CFS File Manager Introduction
- CFS File Manager Overview
- CFS File Manager Operations
- CFS File Manager Commands
- CFS File Manager Telemetry
- CFS File Manager Events
- CFS File Manager Deployment Guide
- CFS File Manager Configuration
- CFS File Manager Table Definitions
- CFS File Manager Operational Constraints
- CFS File Manager Frequently Asked Questions

1.1 CFS File Manager Introduction

Scope

This Doxygen-based User's Guide provides a complete reference for the cFS File Manager (FM) application. The Guide is intended primarily for users of the software (operations personnel, test engineers, and maintenance personnel) who prefer a developer-focused document and direct references to the source code. The "deployment guide" section is intended for mission developers when deploying and configuring the FM application software for a mission flight software build environment.

CFS File Manager Version

Acronyms

Acronym	Description
API	Application Programming Interface
ATP	Absolute Time Processor
ATS	Absolute Time tagged command Sequence
CCSDS	Consultative Committee for Space Data Systems
C&DH	Command and Data Handling
CFE	Core Flight Executive
CFS	Core Flight System
CI	Command Ingest

Cmd	Command
CPU	Central Processing Unit
EDAC	Error Detection and Correction
FDS	Flight Data System
FM	File Manager
FSW	Flight Software
GN&C	Guidance Navigation & Control
GSFC	Goddard Space Flight Center
HK	Housekeeping
HW, H/W	Hardware
ICD	Interface Control Document
ISR	Interrupt Service Routine
OS	Operating System
OSAL	Operating System Abstraction Layer
Pkts	Packets
RAM	Random-Access Memory
RTOS	Real Time Operating System
RTP	Relative Time Processor
RTS	Relative Time tagged command Sequence
SB	Software Bus Service
SBC	Single Board Computer
SC	Stored Commands task
SW, S/W	Software
TBD	To Be Determined
TBL	Table
TLM	Telemetry
UTC	Coordinated Universal Time

1.2 CFS File Manager Overview

The CFS FM application provides onboard file system management services by processing ground commands for copying, moving, and renaming files, decompressing files, creating directories, deleting files and directories, providing file and directory informational telemetry messages, and providing open file and directory listings. The FM software context diagram is shown in Figure FM-1.

Figure 1 Figure FM-1

The FM application interfaces with the Command Ingest (CI) application to receive ground commands. See page CFS File Manager Commands for a detailed description of the FM ground commands. FM utilizes the Operating System Abstraction Layer (OSAL) API library functions for interfacing with the onboard file systems when processing ground commands. The FM application receives messages from the Scheduler (SCH) Application for periodic housekeeping requests. Upon receipt of a housekeeping request, FM will send its housekeeping telemetry packets on the Software Bus. Housekeeping telemetry packets are typically subscribed to by the Housekeeping (HK), Telemetry Output (TO), and Data Storage applications. See page CFS File Manager Telemetry for a detailed description of the contents of the FM housekeeping telemetry packet.

1.3 CFS File Manager Operations

Initialization

Power-On and Processor Resets

Upon a Power-On or Processor reset the FM application initializes all telemetry within its housekeeping telemetry packet. The Command and Command Error counters are set to zero, as well as, all other numerical data within the packet. In addition to initializing telemetry, FM registers for cFE event services, creates its software bus message pipe, and subscribes to FM SB housekeeping requests and commands.

Normal Operation

The CFS FM application is a command and telemetry driven application. FM waits in a command loop infinitely until the software receives a scheduled housekeeping request, ground command, or stored command. Command packets (including housekeeping requests) are processed as they are received. The application sends an FM housekeeping telemetry message on the cFE Software Bus upon receipt of a scheduled housekeeping request produced by the SCH application. This request is typically performed every 4-5 seconds and these housekeeping messages are usually sent to the ground (though this depends on the configuration of the rest of the system).

Various FM ground commands produce telemetry messages as a result of processing the command. These telemetry messages include a directory listing telemetry message, file information telemetry message, and an open file listing telemetry message. See page CFS File Manager Commands for a detailed description of the FM directory listing, file information, and open file listing ground commands. See page CFS File Manager Telemetry for a detailed description of the contents of the FM directory listing, file information, and open file listing telemetry messages.

Other FM ground commands produce files as a result of processing the command. These files include a decompression file, concatenation file, and a directory listing file. See page CFS File Manager Commands for a detailed description of the FM decompress file, concatenate files, and directory listing file ground commands. Note decompression capability is optionally supported via the FM_INCLUDE_DECOMPRESS platform configuration option.

The decompression file is produced using the cFE API function CFE_FS_Decompress which uses code ported from the GNU zip sources. The cFE API function CFE_FS_Decompress produces the decompressed output file. The output file is comparable to that of a file that has been unzipped using the system utility gunzip. A compressed file reduces the size of the file by encoding the file. Compressed files may be used to reduce uplink time to the spacecraft. Decompressing the file will restore the file from its encoded state to the original state of the file.

The concatenation file is a direct copy of the contents of the two command specified source files, with the contents of the first source file specified proceeding the contents of the second source file specified. The contents of the second source file are written with no delineation characters such as spaces or returns following the contents of the first source file.

The directory listing file is written as a binary file and contains a cFE file header at the top of the file contiguously followed by a directory listing status data structure containing an echo of the command specified directory name (OS_MAX_PATH_LEN), directory size in bytes (uint32), total number of file contained in the directory (uint32), and the number of file names written in the directory listing file (uint32). The directory listing is then written contiguously one entry at a time. Each entry in the directory listing includes for each file in the directory, the name of the file(OS_MAX_PATH_LEN), file size in bytes (uint32), and last modification time of the file (uint32). File systems use specific time epochs for their time tagging of files. Since spacecraft systems rarely use an epoch that matches a particular file system, a function is used to convert the file system time (in seconds) to spacecraft time (in seconds). The conversion is controlled by a cFE configuration parameter which is set equal to the number of seconds between the spacecraft's epoch and the file system's epoch. See CFE_FS_Header_t for the cFE file header format. See /FM_DirListFileStat_t for the format of the directory listing status structure. See /FM_DirListFileData_t for the format of a directory listing entry.

1.4 CFS File Manager Commands

CFS File Manager Command Message IDs

CFS File Manager Command Structures

CFS File Manager Command Codes

1.5 CFS File Manager Telemetry

CFS File Manager Telemetry Message IDs

CFS File Manager Telemetry

1.6 CFS File Manager Events

CFS File Manager Event IDs

1.7 CFS File Manager Deployment Guide

To integrate the FM application, follow the CFS App Integration Guide. Also, be sure to set up the configuration parameters and message IDs for the platform

1.8 CFS File Manager Configuration

CFS File Manager Mission Configuration

CFS File Manager Platform Configuration

1.9 CFS File Manager Table Definitions

The File Manager Application defines one table that specifies the file systems contained in your system.

The table contains FM_TABLE_ENTRY_COUNT entries defined by FM_TableEntry_t.

1.10 CFS File Manager Operational Constraints

1. All FM commands that input a file or directory name must use a full path specification.
2. The Decompress File Command should only be used on files that have been compressed.
3. The contents of a directory must be deleted before the directory itself can be deleted.

1.11 CFS File Manager Frequently Asked Questions

(Q) What is the basic flow of the FM application?

The FM application uses the typical CFS application format. At startup, FM performs an initialization sequence and then enters an infinite loop waiting for commands received via the cFE Software Bus. A not so typical feature is the FM child task which is used to process FM commands that might take a very long time to execute.

(Q) In general, what is done during FM initialization?

During initialization, FM performs the common cFE initialization (register for cFE services, create command pipe, subscribe to commands, etc) and then loads the File System Free Space table and creates the FM child task. Errors encountered during the initialization sequence will generally result in the termination of the FM application.

(Q) Why does FM have a child task?

FM commands that process files which may be very large and commands that process directories which may have a very large number of entries can take several seconds to execute. This creates two potential issues: the length of time that FM would be unresponsive to new commands and the length of time that other lower priority tasks would be held off from execution.

Thus, creating a lower priority child task to process "slow" commands allows FM to immediately be ready to receive the next command. And inserting task delays into the child task command process loops provides other lower priority tasks with execution opportunities. Note that inserting delays into the main FM task would have lengthened the time that FM was unavailable for new commands.

(Q) What happens if FM is unable to load the default File System Free Space table during startup initialization?

During initialization, FM must succeed in registering the File System Free Space table with cFE Table Services or the application will terminate. However, it is OK if FM is unable to load the default table data file, although the /FM_GetFreeSpace command will fail until a valid table data file is loaded.

(Q) Is there a limit to the number of files that can be open at one time?

There are two levels of limit to the number of open files. The first limit is defined by the CFE-OSAL configuration element OS_MAX_NUM_OPEN_FILES and must be less than or equal to the second limit which is a value defined in a manner specific to the selected operating system. Note also that certain directory access functions (reading directory entries, etc) require opening the directory and may affect the count of available open files.

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 cfe_version.h header file as part of the API definition; these macros must expand to simple integer values, so that they can be used in simple if directives by the macro preprocessor.

Note the Mission Rev number is provided for missions to be able to identify unique changes they have made to the released software (via clone and own). Specicifally, 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 cfe_version.h for the standard layout and definition of version information. The apps and repositories follow the same pattern by replacing the CFE_ prefix with the appropriate name; for example, osal uses OS_, psp uses CFE_P↩SP_IMPL, and so on.

Suggested pattern for development:

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

Suggested pattern for official releases:

- CFSCOMPONENT_SRC_VERSION: OFFICIAL_GIT_TAG
  - Example: "v7.0.0"
- COMPONENT_VERSION_STRING: "CFSCOMPONENT OFFICIAL RELEASE "CFSCOMPONENT_SRC_VE↩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
STCF	Spacecraft Time Correlation Factor
SW, S/W	Software
TAI	International Atomic Time
TBD	To Be Determined
TBL	Table Services
TID	Task ID
TIME	Time Services
TLM	Telemetry
UTC	Coordinated Universal Time

2.6 cFE Executive Services Overview

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

The interfaces to the ES task include the Ground Interface (commands and telemetry) and the Application Programmer Interfaces (APIs). The ES task interfaces to the OS through the OS Abstraction Layer (OSAL) and platform through the Platform Support Package (PSP).

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

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

- Terminology
- Software Reset
  - Reset Types and Subtypes
  - Exception and Reset (ER) Log
- Application and Child Task Management
  - Starting an Application
  - Stopping an Application
  - Restarting an Application
  - Reloading an Application
  - Listing Current Applications
  - Listing Current Tasks
  - Loading Common Libraries
- Basic File System
- Performance Data Collection
- Critical Data Store

- Memory Pool
- System Log
- Version Identification
- 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	This is the Action the cFE should take if the Application has an exception. • 0 = Do a cFE Processor Reset • Non-Zero = Just restart the Application

Immediately after the cFE completes its initialization, the ES Application first looks for the volatile startup script. The location in the file system is defined by the cFE platform configuration parameter named CFE_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_SPE↩CIAL_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_SU↩BTYPE_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, filename of the object and the application ID. The new application will then show up in the query list downloaded in response to the CFE_ES_QUERY_ALL_CC command.

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

Next: Stopping the Collection of Performance Data
Prev: Performance Data Collection 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_Get↩PoolBuf API) after creating the pool will always cause the memory pool to create a block and return a pointer to the new block. The size of the block depends on the size definitions mentioned earlier. If there is not an exact match between the requested and defined sizes, then the memory pool will create and return the smallest block that meets the following criteria: is a defined size and large enough to hold the request.

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

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

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

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

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

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

The Memory Pool Statistics Telemetry Packet contains the following information:

- **Memory Pool Handle** - the handle, as provided by the operator in the Telemeter Memory Pool Statistics Command. This repeating of the handle in telemetry 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_FIL↩TERS 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 is registered with EVS for filtering, then you have 2 options: 1. You can use the $sc_$cpu_EVS_SetBinFltrMask command using a bit mask of `'0x0000'` which will result in getting all of the events for that Event Id **or** 2. You can remove the registration of that event with EVS (see $sc_$cpu_EVS_DelEvtFltr). Note that option (1) is the preferred method.
(Q) What is the purpose of DEBUG event messages?	
	Event message of type "DEBUG" are primarily used during flight software development in order to provide information that is most likely not needed on orbit. Some commands send debug event messages as verification that a command request was received. When writing the EVS local event log to a file, for example, an event message of type DEBUG is issued. On orbit, this event message is probably not needed. Instead, the command counter is used for command verification.
(Q) How do I find out which events are registered for filtering?	
	EVS provides a command ($sc_$cpu_EVS_WriteAppData2File) which generates a file containing all of the applications that have registered with EVS and all of the filters that are registered for each application. Note that EVS merely generates the file. The file must be transferred to the ground in order to view it.
(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 PipeId) is given back to the caller after the API is executed. Each pipe can be read by only one task, but a task may read more than one pipe. Only the pipe owner is allowed to subscribe to messages on the pipe.

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

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_MEMO↩RY_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?
The memory pool handle is used to analyze the SB memory pool statistics. The cFE ES command (CFE_ES_SEND_MEM_POOL_STATS_CC) to dump the memory pool statistics takes the pool handle as a parameter. These statistics tell how the SB memory pool is configured and gives details on margin. An improperly configured SB memory pool may inhibit communication. This may occur if there is not enough margin to create a block of the size needed for a transfer. Refer to the ES memory pool users guide for more details. Memory Pool
(Q) When sending a message, what message header fields are critical for routing the message?
To route the message properly, the software bus uses only the Message ID and packet length fields from the header of the message. If the packet length field is incorrect, then the buffer allocation for the message will also be incorrect. This may appear to the receiver as a truncated message or a message with unknown data added to the end of the message.
(Q) How many copies of the message are performed in a typical message delivery?
There is a single copy of the message performed when sending a message (from the callers memory space) using CFE_SB_TransmitMsg. When transmitting the message, the software bus copies the message from the callers memory space into a buffer in the software bus memory space. There is also the option to request a buffer from SB, write directly to the buffer and send via CFE_SB_TransmitBuffer. This is equivalent to the previous zero copy implementation. The CFE_SB_ReceiveBuffer API gives the user back a pointer to the buffer. When working with the buffers, the additional complexity to be aware of is the buffer is only available to the app from the request to send (on the sending side), or from the receive until the next receive on the same pipe on the receiving side. If the data is required outside that scope, the app needs a local copy.
(Q) When does the software bus free the buffer during a typical message delivery process? Or how long is the message, and the pointer to the buffer in the CFE_SB_ReceiveBuffer valid?
After receiving a buffer by calling CFE_SB_ReceiveBuffer, the buffer received is valid until the next call to CFE_SB_ReceiveBuffer with the same Pipe Id. If the caller needs the message longer than the next call to CFE_SB_ReceiveBuffer, the caller must copy the message to its memory space.
(Q) The first parameter in the CFE_SB_ReceiveBuffer API is a pointer to a pointer which can get confusing. How can I be sure that the correct address is given for this parameter.
Typically a caller declares a ptr of type CFE_SB_Buffer_t (i.e. CFE_SB_Buffer_t *Ptr) then gives the address of that pointer (&Ptr) as this parameter. After a successful call to CFE_SB_ReceiveBuffer, Ptr will point to the first byte of the software bus buffer. This should be used as a read-only pointer. In systems with an MMU, writes to this pointer may cause a memory protection fault.
(Q) Why am I not seeing expected Message Limit error events or Pipe Overflow events?
It is possible the events are being filtered by cFE Event Services. The filtering for this event may be specified in the platform configuration file or it may have been commanded after the system initializes. There is a corresponding counter for each of these conditions. First verify that the condition is happening by viewing the counter in SB HK telemetry. If the condition is happening, you can view the SB filter information through the EVS App Data Main page by clicking the 'go to' button for SB. The event Id for these events can be learned through a previous event or from the cfe_sb_events.h file.
(Q) Why does the SB provide event filtering through the platform configuration file?
To give the user the ability to filter events before an EVS command can be sent. During system initialization, there are many conditions occurring that can cause a flood of SB events such as No Subscribers, Pipe Overflow and MsgId to Pipe errors. This gives the user a way to limit these events.
(Q) Why does SB have so many debug event messages?
The SB debug messages are positive acknowledgments that an action (like receiving a cmd, creating a pipe or subscribing to a message) has occurred. They are intended to help isolate system problems. For instance, if an expected response to a command is not happening, it may be possible to repeat the scenario with the debug event turned on to verify that the command was successfully received.
(Q) How is the QOS parameter in the CFE_SB_SubscribeEx used by the software bus?

	The QOS parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Setting the QOS 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_SIM↩ ULTANEOUS_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?

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

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

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

What happened?

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

```
TAI = MET + STCF
```

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

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

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

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

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

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

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_TI↩ME_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_TI↩ME_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 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_A↩ DD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECOND↩ S_CC

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

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 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_A↩ DD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECOND↩ S_CC

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

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_A↩ DD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECOND↩ S_CC

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

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_WriteAppInfo2File

Global CFE_ES_QUERY_ALL_TASKS_CC

$sc_$cpu_ES_WriteTaskInfo2File

Global CFE_ES_QUERY_ONE_CC

$sc_$cpu_ES_QueryApp

Global CFE_ES_RELOAD_APP_CC

$sc_$cpu_ES_ReloadApp

Global CFE_ES_RESET_COUNTERS_CC

$sc_$cpu_ES_ResetCtrs

Global CFE_ES_RESET_PR_COUNT_CC

$sc_$cpu_ES_ResetPRCnt

Global CFE_ES_RESTART_APP_CC

$sc_$cpu_ES_ResetApp

Global CFE_ES_RESTART_CC

$sc_$cpu_ES_ProcessorReset, $sc_$cpu_ES_PowerOnReset

Global CFE_ES_SEND_MEM_POOL_STATS_CC

$sc_$cpu_ES_PoolStats

Global CFE_ES_SET_MAX_PR_COUNT_CC

$sc_$cpu_ES_SetMaxPRCnt

Global CFE_ES_SET_PERF_FILTER_MASK_CC

$sc_$cpu_ES_LAFilterMask

Global CFE_ES_SET_PERF_TRIGGER_MASK_CC

$sc_$cpu_ES_LATriggerMask

Global CFE_ES_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_AddSTCFAdj

Global CFE_TIME_ADD_DELAY_CC
$sc_$cpu_TIME_AddClockLat

Global CFE_TIME_NOOP_CC
$sc_$cpu_TIME_NOOP

Global CFE_TIME_RESET_COUNTERS_CC
$sc_$cpu_TIME_ResetCtrs

Global CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
$sc_$cpu_TIME_RequestDiag

Global CFE_TIME_SET_LEAP_SECONDS_CC
$sc_$cpu_TIME_SetClockLeap

Global CFE_TIME_SET_MET_CC
$sc_$cpu_TIME_SetClockMET

Global CFE_TIME_SET_SIGNAL_CC
$sc_$cpu_TIME_SetSignal

Global CFE_TIME_SET_SOURCE_CC
$sc_$cpu_TIME_SetSource

Global CFE_TIME_SET_STATE_CC
$sc_$cpu_TIME_SetState

Global CFE_TIME_SET_STCF_CC
$sc_$cpu_TIME_SetClockSTCF

Global CFE_TIME_SET_TIME_CC
$sc_$cpu_TIME_SetClock

Global CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
$sc_$cpu_TIME_Sub1HzSTCF

Global CFE_TIME_SUB_ADJUST_CC
$sc_$cpu_TIME_SubSTCFAdj

Global CFE_TIME_SUB_DELAY_CC
$sc_$cpu_TIME_SubClockLat

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_PerfFltrMask[MaskCnt]

Global CFE_ES_HousekeepingTlm_Payload::PerfMode

$sc_$cpu_ES_PerfMode

Global CFE_ES_HousekeepingTlm_Payload::PerfState

$sc_$cpu_ES_PerfState

Global CFE_ES_HousekeepingTlm_Payload::PerfTriggerCount

$sc_$cpu_ES_PerfTrigCnt

Global CFE_ES_HousekeepingTlm_Payload::PerfTriggerMask [CFE_MISSION_ES_PERF_MAX_IDS/32]

$sc_$cpu_ES_PerfTrigMask[MaskCnt]

Global CFE_ES_HousekeepingTlm_Payload::ProcessorResets

$sc_$cpu_ES_ProcResetCnt

Global CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion

$sc_$cpu_ES_PSPMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion

$sc_$cpu_ES_PSPMINORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision

$sc_$cpu_ES_PSPMISSIONREV

Global CFE_ES_HousekeepingTlm_Payload::PSPRevision

$sc_$cpu_ES_PSPREVISION

Global CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps

$sc_$cpu_ES_RegCoreApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps

$sc_$cpu_ES_RegExtApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredLibs

$sc_$cpu_ES_RegLibs

Global CFE_ES_HousekeepingTlm_Payload::RegisteredTasks

$sc_$cpu_ES_RegTasks

Global CFE_ES_HousekeepingTlm_Payload::ResetSubtype

$sc_$cpu_ES_ResetSubtype

Global CFE_ES_HousekeepingTlm_Payload::ResetType

$sc_$cpu_ES_ResetType

Global CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed

$sc_$cpu_ES_SYSLOGBYTEUSED

Global CFE_ES_HousekeepingTlm_Payload::SysLogEntries

$sc_$cpu_ES_SYSLOGENTRIES

Global CFE_ES_HousekeepingTlm_Payload::SysLogMode

$sc_$cpu_ES_SYSLOGMODE

Global CFE_ES_HousekeepingTlm_Payload::SysLogSize

$sc_$cpu_ES_SYSLOGSIZE

Global CFE_ES_MemPoolStats::BlockStats [CFE_MISSION_ES_POOL_MAX_BUCKETS]

$sc_$cpu_ES_BlkStats[BLK_SIZES]

Global CFE_ES_MemPoolStats::CheckErrCtr

$sc_$cpu_ES_BlkErrCTR

Global CFE_ES_MemPoolStats::NumBlocksRequested

$sc_$cpu_ES_BlksREQ

Global CFE_ES_MemPoolStats::NumFreeBytes

$sc_$cpu_ES_FreeBytes

Global CFE_ES_MemPoolStats::PoolSize

$sc_$cpu_ES_PoolSize

Global CFE_ES_PoolStatsTlm_Payload::PoolHandle

$sc_$cpu_ES_PoolHandle

Global CFE_EVS_AppTlmData::AppEnableStatus

$sc_$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPENASTAT

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].APPMSGSENTC

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_LOGOVERFLOWC

Global CFE_EVS_HousekeepingTlm_Payload::MessageFormatMode

$sc_$cpu_EVS_MSGFMTMODE

Global CFE_EVS_HousekeepingTlm_Payload::MessageSendCounter

$sc_$cpu_EVS_MSGSENTC

Global CFE_EVS_HousekeepingTlm_Payload::MessageTruncCounter

$sc_$cpu_EVS_MSGTRUNC

Global CFE_EVS_HousekeepingTlm_Payload::OutputPort

$sc_$cpu_EVS_OUTPUTPORT

Global CFE_EVS_HousekeepingTlm_Payload::Spare1

$sc_$cpu_EVS_HK_SPARE1

Global CFE_EVS_HousekeepingTlm_Payload::Spare2

$sc_$cpu_EVS_HK_SPARE2

Global CFE_EVS_HousekeepingTlm_Payload::Spare3

$sc_$cpu_EVS_HK_SPARE3

Global CFE_EVS_HousekeepingTlm_Payload::UnregisteredAppCounter

$sc_$cpu_EVS_UNREGAPPC

Global CFE_EVS_LongEventTlm_Payload::Message [CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]

$sc_$cpu_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::PipeId

$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDPIPEID

Global CFE_SB_PipeDepthStats::Spare

$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES].SB_PDSPARE

Global CFE_SB_StatsTlm_Payload::MaxMemAllowed

$sc_$cpu_SB_Stat.SB_SMMBMALW

Global CFE_SB_StatsTlm_Payload::MaxMsgIdsAllowed

$sc_$cpu_SB_Stat.SB_SMMMIDALW

Global CFE_SB_StatsTlm_Payload::MaxPipeDepthAllowed

$sc_$cpu_SB_Stat.SB_SMMPDALW

Global CFE_SB_StatsTlm_Payload::MaxPipesAllowed

$sc_$cpu_SB_Stat.SB_SMMPALW

Global CFE_SB_StatsTlm_Payload::MaxSubscriptionsAllowed

$sc_$cpu_SB_Stat.SB_SMMSALW

Global CFE_SB_StatsTlm_Payload::MemInUse

$sc_$cpu_SB_Stat.SB_SMBMIU

Global CFE_SB_StatsTlm_Payload::MsgIdsInUse

$sc_$cpu_SB_Stat.SB_SMMIDIU

Global CFE_SB_StatsTlm_Payload::PeakMemInUse

$sc_$cpu_SB_Stat.SB_SMPBMIU

Global CFE_SB_StatsTlm_Payload::PeakMsgIdsInUse

$sc_$cpu_SB_Stat.SB_SMPMIDIU

Global CFE_SB_StatsTlm_Payload::PeakPipesInUse

$sc_$cpu_SB_Stat.SB_SMPPIU

Global CFE_SB_StatsTlm_Payload::PeakSBBuffersInUse

$sc_$cpu_SB_Stat.SB_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_ByteAlignPad1

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_DLatentS, $sc_$cpu_TIME_DLatentSs

Global CFE_TIME_DiagnosticTlm_Payload::AtToneLatch

$sc_$cpu_TIME_DTValidS, $sc_$cpu_TIME_DTValidSs

Global CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds

$sc_$cpu_TIME_DLeapS

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_DAPIState

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_DLocalS, $sc_$cpu_TIME_DLocalSs

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_DLatentDir

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_DTDSs

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_DTTSs

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_DLogicalMET

Global CFE_TIME_HousekeepingTlm_Payload::ClockStateAPI

$sc_$cpu_TIME_DAPIState

Global CFE_TIME_HousekeepingTlm_Payload::ClockStateFlags

$sc_$cpu_TIME_StateFlg, $sc_$cpu_TIME_FlagSet, $sc_$cpu_TIME_FlagFly, $sc_$cpu_TIME_FlagSrc, $sc_↩ $cpu_TIME_FlagPri, $sc_$cpu_TIME_FlagSfly, $sc_$cpu_TIME_FlagCfly, $sc_$cpu_TIME_FlagAdjd, $sc_$cpu↩ _TIME_Flag1Hzd, $sc_$cpu_TIME_FlagClat, $sc_$cpu_TIME_FlagSorC, $sc_$cpu_TIME_FlagNIU

Global CFE_TIME_HousekeepingTlm_Payload::CommandCounter

$sc_$cpu_TIME_CMDPC

Global CFE_TIME_HousekeepingTlm_Payload::CommandErrorCounter

$sc_$cpu_TIME_CMDEC

Global CFE_TIME_HousekeepingTlm_Payload::LeapSeconds

$sc_$cpu_TIME_LeapSecs

Global CFE_TIME_HousekeepingTlm_Payload::Seconds1HzAdj

$sc_$cpu_TIME_1HzAdjSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsDelay

$sc_$cpu_TIME_1HzAdjSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsMET

$sc_$cpu_TIME_METSecs

Global CFE_TIME_HousekeepingTlm_Payload::SecondsSTCF

$sc_$cpu_TIME_STCFSecs

Global CFE_TIME_HousekeepingTlm_Payload::Subsecs1HzAdj

$sc_$cpu_TIME_1HzAdjSSecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsDelay

$sc_$cpu_TIME_1HzAdjSSecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsMET

$sc_$cpu_TIME_METSubsecs

Global CFE_TIME_HousekeepingTlm_Payload::SubsecsSTCF

$sc_$cpu_TIME_STCFSubsecs

3 Glossary of Terms

Term	Definition
Application (or App)	A set of data and functions that is treated as a single entity by the cFE. cFE resources are allocated on a per-Application basis. Applications are made up of a Main Task and zero or more Child Tasks.
Application ID	A processor unique reference to an Application. **NOTE: This is different from a CCSDS Application ID which is referred to as an "APID."**
Application Programmer's Interface (API)	A set of routines, protocols, and tools for building software applications
Platform Support Package (PSP)	A collection of user-provided facilities that interface an OS and the cFE with a specific hardware platform. The PSP is responsible for hardware initialization.
Child Task	A separate thread of execution that is spawned by an Application's Main Task.
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_Event↩ Type_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_E↩S_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_BackgroundFileDumpIsPending - 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_ResourceId_ToInteger - Convert a resource ID to an integer.
  - CFE_ResourceId_FromInteger - Convert an integer to a resource ID.
  - CFE_ResourceId_Equal - Compare two Resource ID values for equality.
  - CFE_ResourceId_IsDefined - Check if a resource ID value is defined.
  - CFE_ResourceId_GetBase - Get the Base value (type/category) from a resource ID value.
  - CFE_ResourceId_GetSerial - Get the Serial Number (sequential ID) from a resource ID value.
  - CFE_ResourceId_FindNext - Locate the next resource ID which does not map to an in-use table entry.
  - CFE_ResourceId_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_PipeId_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_GetPipeIdByName - 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 it’s 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_B↩ ASED 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 File Manager Mission Configuration 131

CFS File Manager Command Message IDs 132

CFS File Manager Telemetry Message IDs 133

CFS File Manager Platform Configuration 134

CFS File Manager Event IDs 146

CFS File Manager Command Structures 276

CFS File Manager Telemetry 277

CFS File Manager Command Codes 278

CFS File Manager Version 301

cFE Return Code Defines 302

cFE Resource ID APIs 339

cFE Entry/Exit APIs 343

cFE Application Control APIs 345

cFE Application Behavior APIs 348

cFE Information APIs 353

cFE Child Task APIs 363

cFE Miscellaneous APIs 368

cFE Critical Data Store APIs 372

cFE Memory Manager APIs 377

cFE Performance Monitor APIs 385

cFE Generic Counter APIs 388

cFE Registration APIs 396

cFE Send Event APIs 398

cFE Reset Event Filter APIs 403

cFE File Header Management APIs 405

cFE File Utility APIs 410

cFE Generic Message APIs 416

cFE Message Primary Header APIs 417

cFE Message Extended Header APIs 428

cFE Message Secondary Header APIs 435

cFE Message Id APIs 441

cFE Pipe Management APIs 444

cFE Message Subscription Control APIs 451

cFE Send/Receive Message APIs 457

cFE Zero Copy APIs 460

cFE Message Characteristics APIs 463

cFE Message ID APIs 468

cFE SB Pipe options 471

cFE Registration APIs 472

cFE Manage Table Content APIs 478

cFE Access Table Content APIs 485

cFE Get Table Information APIs 490

cFE Table Type Defines 494

cFE Get Current Time APIs 498

cFE Get Time Information APIs 502

cFE Time Arithmetic APIs 505

cFE Time Conversion APIs 508

cFE External Time Source APIs 511

cFE Miscellaneous Time APIs 516

cFE Resource ID base values 518

cFE Clock State Flag Defines 520

OSAL Semaphore State Defines 523

OSAL Binary Semaphore APIs 524

OSAL BSP low level access APIs 530

OSAL Real Time Clock APIs 531

OSAL Core Operation APIs 543

OSAL Counting Semaphore APIs 547

OSAL Directory APIs 553

OSAL Return Code Defines 558

OSAL Error Info APIs 569

OSAL File Access Option Defines 571

OSAL Reference Point For Seek Offset Defines 572

OSAL Standard File APIs 573

OSAL File System Level APIs 586

OSAL Heap APIs 595

OSAL Object Type Defines 596

OSAL Object ID Utility APIs 600

OSAL Dynamic Loader and Symbol APIs 607

OSAL Mutex APIs 612

OSAL Network ID APIs 617

OSAL Printf APIs 619

OSAL Message Queue APIs 621

OSAL Select APIs 626

OSAL Shell APIs 631

OSAL Socket Address APIs 632

OSAL Socket Management APIs 636

OSAL Task APIs 644

OSAL Time Base APIs 651

OSAL Timer APIs 657

8 Data Structure Index

8.1 Data Structures

Here are the data structures with brief descriptions:

CCSDS_ExtendedHeader
CCSDS packet extended header 664

CCSDS_PrimaryHeader
CCSDS packet primary header 665

CFE_ES_AppInfo
Application Information 666

CFE_ES_AppNameCmd
Generic application name command 672

CFE_ES_AppNameCmd_Payload
Generic application name command payload 673

CFE_ES_AppReloadCmd_Payload
Reload Application Command Payload 674

CFE_ES_BlockStats
Block statistics 675

CFE_ES_CDSRegDumpRec
CDS Register Dump Record 676

CFE_ES_DeleteCDSCmd
Delete Critical Data Store Command 678

CFE_ES_DeleteCDSCmd_Payload
Delete Critical Data Store Command Payload 679

CFE_ES_DumpCDSRegistryCmd
Dump CDS Registry Command 680

CFE_ES_DumpCDSRegistryCmd_Payload
Dump CDS Registry Command Payload 681

CFE_ES_FileNameCmd
Generic file name command 682

CFE_ES_FileNameCmd_Payload
Generic file name command payload 683

CFE_ES_HousekeepingTlm 684

CFE_ES_HousekeepingTlm_Payload 684

CFE_ES_MemPoolStats
Memory Pool Statistics 697

CFE_ES_MemStatsTlm 699

CFE_ES_NoArgsCmd
Generic "no arguments" command 700

CFE_ES_OneAppTlm 701

CFE_ES_OneAppTlm_Payload 702

CFE_ES_OverWriteSysLogCmd
Overwrite/Discard System Log Configuration Command Payload 703

CFE_ES_OverWriteSysLogCmd_Payload
Overwrite/Discard System Log Configuration Command Payload 704

CFE_ES_PoolAlign
Pool Alignment 705

CFE_ES_PoolStatsTlm_Payload 706

CFE_ES_ReloadAppCmd
Reload Application Command 707

CFE_ES_RestartCmd
Restart cFE Command 708

CFE_ES_RestartCmd_Payload
Restart cFE Command Payload 709

CFE_ES_SendMemPoolStatsCmd
Send Memory Pool Statistics Command 710

CFE_ES_SendMemPoolStatsCmd_Payload
Send Memory Pool Statistics Command Payload 711

CFE_ES_SetMaxPRCountCmd
Set Maximum Processor Reset Count Command 712

CFE_ES_SetMaxPRCountCmd_Payload
Set Maximum Processor Reset Count Command Payload 713

CFE_ES_SetPerfFilterMaskCmd
Set Performance Analyzer Filter Mask Command 713

CFE_ES_SetPerfFilterMaskCmd_Payload
Set Performance Analyzer Filter Mask Command Payload 714

CFE_ES_SetPerfTriggerMaskCmd
Set Performance Analyzer Trigger Mask Command 715

CFE_ES_SetPerfTrigMaskCmd_Payload
Set Performance Analyzer Trigger Mask Command Payload 716

CFE_ES_StartApp
Start Application Command 717

CFE_ES_StartAppCmd_Payload
Start Application Command Payload 718

CFE_ES_StartPerfCmd_Payload
Start Performance Analyzer Command Payload 720

CFE_ES_StartPerfDataCmd
Start Performance Analyzer Command 721

CFE_ES_StopPerfCmd_Payload
Stop Performance Analyzer Command Payload 722

CFE_ES_StopPerfDataCmd
Stop Performance Analyzer Command 723

CFE_ES_TaskInfo
Task Information 724

CFE_EVS_AppDataCmd_Payload
Write Event Services Application Information to File Command Payload 727

CFE_EVS_AppNameBitMaskCmd
Generic App Name and Bitmask Command 727

CFE_EVS_AppNameBitMaskCmd_Payload
Generic App Name and Bitmask Command Payload 728

CFE_EVS_AppNameCmd
Generic App Name Command 730

CFE_EVS_AppNameCmd_Payload
Generic App Name Command Payload 731

CFE_EVS_AppNameEventIDCmd
Generic App Name and Event ID Command 732

CFE_EVS_AppNameEventIDCmd_Payload
Generic App Name and Event ID Command Payload 733

CFE_EVS_AppNameEventIDMaskCmd
Generic App Name, Event ID, Mask Command 734

CFE_EVS_AppNameEventIDMaskCmd_Payload
Generic App Name, Event ID, Mask Command Payload 735

CFE_EVS_AppTlmData 736

CFE_EVS_BinFilter
Event message filter definition structure 737

CFE_EVS_BitMaskCmd
Generic Bitmask Command 738

CFE_EVS_BitMaskCmd_Payload
Generic Bitmask Command Payload 739

CFE_EVS_HousekeepingTlm 740

CFE_EVS_HousekeepingTlm_Payload 741

CFE_EVS_LogFileCmd_Payload
Write Event Log to File Command Payload 746

CFE_EVS_LongEventTlm 747

CFE_EVS_LongEventTlm_Payload 748

CFE_EVS_NoArgsCmd
Command with no additional arguments 750

CFE_EVS_PacketID 750

CFE_EVS_SetEventFormatCode_Payload
Set Event Format Mode Command Payload 752

CFE_EVS_SetEventFormatModeCmd
Set Event Format Mode Command 753

CFE_EVS_SetLogMode_Payload
Set Log Mode Command Payload 754

CFE_EVS_SetLogModeCmd
Set Log Mode Command 755

CFE_EVS_ShortEventTlm 756

CFE_EVS_ShortEventTlm_Payload 757

CFE_EVS_WriteAppDataFileCmd
Write Event Services Application Information to File Command 758

CFE_EVS_WriteLogDataFileCmd
Write Event Log to File Command 759

CFE_FS_FileWriteMetaData
External Metadata/State object associated with background file writes 760

CFE_FS_Header
Standard cFE File header structure definition 762

CFE_SB_AllSubscriptionsTlm 764

CFE_SB_AllSubscriptionsTlm_Payload 765

CFE_SB_HousekeepingTlm 767

CFE_SB_HousekeepingTlm_Payload 768

CFE_SB_Msg
Software Bus generic message 773

CFE_SB_MsgId_t
CFE_SB_MsgId_t type definition 774

CFE_SB_MsgMapFileEntry
SB Map File Entry 775

CFE_SB_PipeDepthStats
SB Pipe Depth Statistics 776

CFE_SB_PipeInfoEntry
SB Pipe Information File Entry 778

CFE_SB_Qos_t
Quality Of Service Type Definition 781

CFE_SB_RouteCmd
Enable/Disable Route Command 782

CFE_SB_RouteCmd_Payload
Enable/Disable Route Command Payload 783

CFE_SB_RoutingFileEntry
SB Routing File Entry 785

CFE_SB_SingleSubscriptionTlm 787

CFE_SB_SingleSubscriptionTlm_Payload 787

CFE_SB_StatsTlm 789

CFE_SB_StatsTlm_Payload 790

CFE_SB_SubEntries
SB Previous Subscriptions Entry 795

CFE_SB_WriteFileInfoCmd
Write File Info Command 797

CFE_SB_WriteFileInfoCmd_Payload
Write File Info Command Payload 798

CFE_TBL_AbortLoadCmd
Abort Load Command 798

CFE_TBL_AbortLoadCmd_Payload
Abort Load Command Payload 799

CFE_TBL_ActivateCmd
Activate Table Command 800

CFE_TBL_ActivateCmd_Payload
Activate Table Command Payload 801

CFE_TBL_DelCDSCmd_Payload
Delete Critical Table CDS Command Payload 802

CFE_TBL_DeleteCDSCmd
Delete Critical Table CDS Command 803

CFE_TBL_DumpCmd 804

CFE_TBL_DumpCmd_Payload
Dump Table Command Payload 805

CFE_TBL_DumpRegistryCmd
Dump Registry Command 806

CFE_TBL_DumpRegistryCmd_Payload
Dump Registry Command Payload 807

CFE_TBL_File_Hdr
The definition of the header fields that are included in CFE Table Data files 808

CFE_TBL_FileDef 810

CFE_TBL_HousekeepingTlm 811

CFE_TBL_HousekeepingTlm_Payload 812

CFE_TBL_Info
Table Info 818

CFE_TBL_LoadCmd
Load Table Command 822

CFE_TBL_LoadCmd_Payload
Load Table Command Payload 823

CFE_TBL_NoArgsCmd
Generic "no arguments" command 824

CFE_TBL_NotifyCmd 824

CFE_TBL_NotifyCmd_Payload
Table Management Notification Command Payload 825

CFE_TBL_SendRegistryCmd
Send Table Registry Command 826

CFE_TBL_SendRegistryCmd_Payload
Send Table Registry Command Payload 827

CFE_TBL_TableRegistryTlm 828

CFE_TBL_TblRegPacket_Payload 829

CFE_TBL_ValidateCmd
Validate Table Command 835

CFE_TBL_ValidateCmd_Payload
Validate Table Command Payload 836

CFE_TIME_DiagnosticTlm 837

CFE_TIME_DiagnosticTlm_Payload 837

CFE_TIME_HousekeepingTlm 850

CFE_TIME_HousekeepingTlm_Payload 851

CFE_TIME_LeapsCmd_Payload
Set leap seconds command payload 855

CFE_TIME_NoArgsCmd
Generic no argument command 856

CFE_TIME_OneHzAdjustmentCmd
Generic seconds, subseconds adjustment command 857

CFE_TIME_OneHzAdjustmentCmd_Payload
Generic seconds, subseconds command payload 858

CFE_TIME_SetLeapSecondsCmd
Set leap seconds command 859

CFE_TIME_SetSignalCmd
Set tone signal source command 860

CFE_TIME_SetSourceCmd
Set time data source command 861

CFE_TIME_SetStateCmd
Set clock state command 862

CFE_TIME_SignalCmd_Payload
Set tone signal source command payload 863

CFE_TIME_SourceCmd_Payload
Set time data source command payload 864

CFE_TIME_StateCmd_Payload
Set clock state command payload 864

CFE_TIME_SysTime
Data structure used to hold system time values 865

CFE_TIME_TimeCmd
Generic seconds, microseconds argument command 866

CFE_TIME_TimeCmd_Payload
Generic seconds, microseconds command payload 867

CFE_TIME_ToneDataCmd
Time at tone data command 868

CFE_TIME_ToneDataCmd_Payload
Time at tone data command payload 869

FM_ChildQueueEntry_t
Child Task Interface command queue entry structure 871

FM_ConcatCmd_t
Concatenate Files command packet structure 874

FM_CopyFileCmd_t
Copy File command packet structure 876

FM_CreateDirCmd_t
Create Directory command packet structure 878

FM_DecompressCmd_t
Decompress File command packet structure 879

FM_DeleteAllCmd_t
Delete All command packet structure 880

FM_DeleteDirCmd_t
Delete Directory command packet structure 881

FM_DeleteFileCmd_t
Delete File command packet structure 882

FM_DirListEntry_t
Get Directory Listing entry structure 883

FM_DirListFileStats_t
Get Directory Listing file statistics structure 885

FM_DirListPkt_t
Get Directory Listing telemetry packet 886

FM_FileInfoPkt_t
Get File Info telemetry packet 888

FM_FreeSpacePkt_t
Get Free Space telemetry packet 891

FM_FreeSpacePktEntry_t
Get Free Space list entry structure 892

FM_FreeSpaceTable_t
Get Free Space table definition 893

FM_GetDirFileCmd_t
Get DIR List to File command packet structure 894

FM_GetDirPktCmd_t
Get DIR List to Packet command packet structure 896

FM_GetFileInfoCmd_t
Get File Info command packet structure 898

FM_GetFreeSpaceCmd_t
Get Free Space command packet structure 899

FM_GetOpenFilesCmd_t
Get Open Files command packet structure 900

FM_GlobalData_t
Application global data structure 901

FM_HousekeepingCmd_t
Housekeeping Request command packet structure 910

FM_HousekeepingPkt_t
Housekeeping telemetry packet 911

FM_MoveFileCmd_t
Move File command packet structure 914

FM_NoopCmd_t
No-Operation command packet structure 916

FM_OpenFilesEntry_t
Get Open Files list entry structure 917

FM_OpenFilesPkt_t
Get Open Files telemetry packet 918

FM_RenameFileCmd_t
Rename File command packet structure 919

FM_ResetCmd_t
Reset Counters command packet structure 920

FM_SetPermCmd_t
Set Permissions for a file 921

FM_SetTableStateCmd_t
Set Table State command packet structure 922

FM_TableEntry_t
Get Free Space table entry 924

OS_bin_sem_prop_t
OSAL binary semaphore properties 925

OS_count_sem_prop_t
OSAL counting semaphore properties 926

os_dirent_t
Directory entry 927

OS_FdSet
An abstract structure capable of holding several OSAL IDs 928

OS_file_prop_t
OSAL file properties 928

os_fsinfo_t
OSAL file system info 930

os_fstat_t
File system status 931

OS_heap_prop_t
OSAL heap properties 932

OS_module_address_t
OSAL module address properties 933

OS_module_prop_t
OSAL module properties 935

OS_mut_sem_prop_t
OSAL mutex properties 937

OS_queue_prop_t
OSAL queue properties 938

OS_SockAddr_t
Encapsulates a generic network address 938

OS_SockAddrData_t
Storage buffer for generic network address 939

OS_socket_prop_t
Encapsulates socket properties 941

OS_static_symbol_record_t
Associates a single symbol name with a memory address 942

OS_statvfs_t 943

OS_task_prop_t
OSAL task properties 944

OS_time_t
OSAL time interval structure 945

OS_timebase_prop_t
Time base properties 946

OS_timer_prop_t
Timer properties 948

9 File Index

9.1 File List

Here is a list of all files with brief descriptions:

apps/fm/fsw/mission_inc/fm_perfids.h 949

apps/fm/fsw/platform_inc/fm_msgids.h 949

apps/fm/fsw/platform_inc/fm_platform_cfg.h 950

apps/fm/fsw/src/fm_app.c 951

apps/fm/fsw/src/fm_app.h 962

apps/fm/fsw/src/fm_child.c 972

apps/fm/fsw/src/fm_child.h 1000

apps/fm/fsw/src/fm_cmd_utils.c 1028

apps/fm/fsw/src/fm_cmd_utils.h 1048

apps/fm/fsw/src/fm_cmds.c 1066

apps/fm/fsw/src/fm_cmds.h 1090

apps/fm/fsw/src/fm_defs.h 1114

apps/fm/fsw/src/fm_events.h 1117

apps/fm/fsw/src/fm_msg.h 1129

apps/fm/fsw/src/fm_msgdefs.h 1132

apps/fm/fsw/src/fm_tbl.c 1133

apps/fm/fsw/src/fm_tbl.h 1137

apps/fm/fsw/src/fm_verify.h 1142

apps/fm/fsw/src/fm_version.h 1143

apps/fm/fsw/tables/fm_freespace.c 1143

build/docs/osconfig-example.h 1144

cfe/cmake/sample_defs/cpu1_msgids.h 1153

cfe/cmake/sample_defs/cpu1_platform_cfg.h 1161

cfe/cmake/sample_defs/sample_mission_cfg.h 1218

cfe/cmake/sample_defs/sample_perfids.h 1239

cfe/modules/core_api/fsw/inc/cfe.h 1243

cfe/modules/core_api/fsw/inc/cfe_config.h 1243

cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h 1247

cfe/modules/core_api/fsw/inc/cfe_endian.h 1248

cfe/modules/core_api/fsw/inc/cfe_error.h 1249

cfe/modules/core_api/fsw/inc/cfe_es.h 1258

cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h 1262

cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h 1269

cfe/modules/core_api/fsw/inc/cfe_evs.h 1281

cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h 1283

cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h 1286

cfe/modules/core_api/fsw/inc/cfe_fs.h 1291

cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h 1292

cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h 1295

cfe/modules/core_api/fsw/inc/cfe_msg.h 1298

cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h 1300

cfe/modules/core_api/fsw/inc/cfe_resourceid.h 1308

cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h 1315

cfe/modules/core_api/fsw/inc/cfe_sb.h 1317

cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h 1319

cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h 1324

cfe/modules/core_api/fsw/inc/cfe_tbl.h 1328

cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h 1329

cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h 1332

cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h 1334

cfe/modules/core_api/fsw/inc/cfe_time.h 1336

cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h 1338

cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.h 1340

cfe/modules/core_api/fsw/inc/cfe_version.h 1346

cfe/modules/es/fsw/inc/cfe_es_events.h 1349

cfe/modules/es/fsw/inc/cfe_es_msg.h 1382

cfe/modules/evs/fsw/inc/cfe_evs_events.h 1418

cfe/modules/evs/fsw/inc/cfe_evs_msg.h 1432

cfe/modules/msg/fsw/inc/ccsds_hdr.h 1464

cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h 1465

cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h 1466

cfe/modules/sb/fsw/inc/cfe_sb_events.h 1468

cfe/modules/sb/fsw/inc/cfe_sb_msg.h 1492

cfe/modules/tbl/fsw/inc/cfe_tbl_events.h 1510

cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h 1536

cfe/modules/time/fsw/inc/cfe_time_events.h 1553

cfe/modules/time/fsw/inc/cfe_time_msg.h 1567

osal/src/os/inc/common_types.h 1593

osal/src/os/inc/osapi-binsem.h 1601

osal/src/os/inc/osapi-bsp.h 1602

osal/src/os/inc/osapi-clock.h 1603

osal/src/os/inc/osapi-common.h 1605

osal/src/os/inc/osapi-constants.h 1608

osal/src/os/inc/osapi-countsem.h 1609

osal/src/os/inc/osapi-dir.h 1610

osal/src/os/inc/osapi-error.h 1611

osal/src/os/inc/osapi-file.h 1615

osal/src/os/inc/osapi-filesys.h 1619

osal/src/os/inc/osapi-heap.h 1621

osal/src/os/inc/osapi-idmap.h 1621

osal/src/os/inc/osapi-macros.h 1623

osal/src/os/inc/osapi-module.h 1626

osal/src/os/inc/osapi-mutex.h 1627

osal/src/os/inc/osapi-network.h 1628

osal/src/os/inc/osapi-printf.h 1629

osal/src/os/inc/osapi-queue.h 1629

osal/src/os/inc/osapi-select.h 1630

osal/src/os/inc/osapi-shell.h 1631

osal/src/os/inc/osapi-sockets.h 1631

osal/src/os/inc/osapi-task.h 1634

osal/src/os/inc/osapi-timebase.h 1638

osal/src/os/inc/osapi-timer.h 1639

osal/src/os/inc/osapi-version.h 1640

osal/src/os/inc/osapi.h 1645

psp/fsw/inc/cfe_psp.h 1646

10 Module Documentation

10.1 CFS File Manager Mission Configuration

Macros

- #define FM_APPMAIN_PERF_ID 39

  Main application performance ID.
- #define FM_CHILD_TASK_PERF_ID 44

  Child task performance ID.

10.1.1 Detailed Description

10.1.2 Macro Definition Documentation

10.1.2.1 FM_APPMAIN_PERF_ID

```
#define FM_APPMAIN_PERF_ID 39
```

Main application performance ID.

Definition at line 32 of file fm_perfids.h.

Referenced by FM_AppMain().

10.1.2.2 FM_CHILD_TASK_PERF_ID

```
#define FM_CHILD_TASK_PERF_ID 44
```

Child task performance ID.

Definition at line 33 of file fm_perfids.h.

Referenced by FM_ChildConcatCmd(), FM_ChildFileInfoCmd(), FM_ChildLoop(), and FM_ChildSleepStat().

10.2 CFS File Manager Command Message IDs

Macros

- #define FM_CMD_MID 0x188C /** < \brief FM ground commands */
- #define FM_SEND_HK_MID 0x188D /** < \brief FM send housekeeping */

10.2.1 Detailed Description

10.2.2 Macro Definition Documentation

10.2.2.1 FM_CMD_MID

```
#define FM_CMD_MID 0x188C /** < \brief FM ground commands */
```

Definition at line 33 of file fm_msgids.h.

Referenced by FM_AppInit(), and FM_ProcessPkt().

10.2.2.2 FM_SEND_HK_MID

```
#define FM_SEND_HK_MID 0x188D /** < \brief FM send housekeeping */
```

Definition at line 34 of file fm_msgids.h.

Referenced by FM_AppInit(), and FM_ProcessPkt().

10.3 CFS File Manager Telemetry Message IDs

Macros

- #define FM_HK_TLM_MID 0x088A /** < \brief FM housekeeping */
- #define FM_FILE_INFO_TLM_MID 0x088B /** < \brief FM get file info */
- #define FM_DIR_LIST_TLM_MID 0x088C /** < \brief FM get dir list */
- #define FM_OPEN_FILES_TLM_MID 0x088D /** < \brief FM get open files */
- #define FM_FREE_SPACE_TLM_MID 0x088E /** < \brief FM get free space */

10.3.1 Detailed Description

10.3.2 Macro Definition Documentation

10.3.2.1 FM_DIR_LIST_TLM_MID

```
#define FM_DIR_LIST_TLM_MID 0x088C /** < \brief FM get dir list */
```

Definition at line 45 of file fm_msgids.h.

Referenced by FM_ChildDirListPktCmd().

10.3.2.2 FM_FILE_INFO_TLM_MID

```
#define FM_FILE_INFO_TLM_MID 0x088B /** < \brief FM get file info */
```

Definition at line 44 of file fm_msgids.h.

Referenced by FM_ChildFileInfoCmd().

10.3.2.3 FM_FREE_SPACE_TLM_MID

```
#define FM_FREE_SPACE_TLM_MID 0x088E /** < \brief FM get free space */
```

Definition at line 47 of file fm_msgids.h.

Referenced by FM_GetFreeSpaceCmd().

10.3.2.4 FM_HK_TLM_MID

```
#define FM_HK_TLM_MID 0x088A /** < \brief FM housekeeping */
```

Definition at line 43 of file fm_msgids.h.

Referenced by FM_ReportHK().

10.3.2.5 FM_OPEN_FILES_TLM_MID

```
#define FM_OPEN_FILES_TLM_MID 0x088D /** < \brief FM get open files */
```

Definition at line 46 of file fm_msgids.h.

Referenced by FM_GetOpenFilesCmd().

10.4 CFS File Manager Platform Configuration

Macros

- #define FM_APP_NAME "FM"

  File Manager Application Name.
- #define FM_APP_PIPE_NAME "FM_CMD_PIPE"

  File Manager Command Pipe Name.
- #define FM_APP_PIPE_DEPTH 10

  File Manager Command Pipe Depth.
- #define FM_MISSION_REV 0

  Mission specific version number for FM application.
- #define FM_DIR_LIST_FILE_DEFNAME "/ram/fm_dirlist.out"

  Default Directory List Output Filename.
- #define FM_DIR_LIST_FILE_ENTRIES 3000

  Maximum Directory List Output File Entries.
- #define FM_DIR_LIST_FILE_SUBTYPE 12345

  Directory List Output File Header Sub-Type.
- #define FM_DIR_LIST_PKT_ENTRIES 20

  Directory List Telemetry Packet Entry Count.
- #define FM_CHILD_FILE_BLOCK_SIZE 2048

  Child Task File I/O Control Settings.
- #define FM_CHILD_FILE_LOOP_COUNT 16
- #define FM_CHILD_FILE_SLEEP_MS 20
- #define FM_CHILD_STAT_SLEEP_MS 0

  Child file stat sleep.
- #define FM_CHILD_STAT_SLEEP_FILECOUNT 0
- #define FM_CHILD_QUEUE_DEPTH 3

  Child Task Command Queue Entry Count.
- #define FM_CHILD_TASK_NAME "FM_CHILD_TASK"

  Child Task Name - cFE object name.
- #define FM_CHILD_TASK_STACK_SIZE 20480

  Child Task Stack Size.
- #define FM_CHILD_TASK_PRIORITY 205

  Child Task Execution Priority.
- #define FM_CHILD_SEM_NAME "FM_CHILD_SEM"

  Child Task Semaphore Name - cFE object name.
- #define FM_TABLE_CFE_NAME "FreeSpace"

  Free Space Table Name - cFE object name.
- #define FM_TABLE_DEF_NAME "/cf/fm_freespace.tbl"

  Free Space Table Name - filename with path.
- #define FM_TABLE_FILENAME "fm_freespace.tbl"

  Free Space Table Name - filename without path.
- #define FM_TABLE_DEF_DESC "FM File System Free Space Table"

  Free Space Table Description.
- #define FM_TABLE_ENTRY_COUNT 8

  Number of Free Space Table Entries.
- #define FM_TABLE_VALIDATION_ERR (-1)

  Table Data Validation Error Code.

10.4.1 Detailed Description

10.4.2 Macro Definition Documentation

10.4.2.1 FM_APP_NAME

```
#define FM_APP_NAME "FM"
```

File Manager Application Name.

Description:

This definition must match the name used at startup by the cFE Executive Services when creating the FM 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:

FM 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 54 of file fm_platform_cfg.h.

10.4.2.2 FM_APP_PIPE_DEPTH

```
#define FM_APP_PIPE_DEPTH 10
```

File Manager Command Pipe Depth.

Description:

This definition sets the total number of packets that may queue in the FM command pipe. The limit for individual message types in the queue is the default cFE Software Bus subscription limit.

Limits:

It is recommended that this value be no less than 4 and no greater than 20 packets, but this is not enforced by FM.

Definition at line 83 of file fm_platform_cfg.h.

Referenced by FM_AppInit().

10.4.2.3 FM_APP_PIPE_NAME

```
#define FM_APP_PIPE_NAME "FM_CMD_PIPE"
```

File Manager Command Pipe Name.

Description:

This definition is the name used at startup when creating a cFE Software Bus command pipe for the FM application.

Limits:

FM requires that this name be defined, but otherwise places no limits on the definition. Refer to CFE Software Bus Services for specific information on limits related to pipe names.

Definition at line 68 of file fm_platform_cfg.h.

Referenced by FM_AppInit().

10.4.2.4 FM_CHILD_FILE_BLOCK_SIZE

```
#define FM_CHILD_FILE_BLOCK_SIZE 2048
```

Child Task File I/O Control Settings.

Description:

These definitions control the amount of file data that the FM child task will process before giving up the CPU to allow other tasks time to run.

FM_CHILD_FILE_BLOCK_SIZE defines the size of each block of file data that the FM child task will read or write. This value also defines the size of the FM child task I/O buffer that exists in global memory.

FM_CHILD_FILE_LOOP_COUNT defines the number of file data blocks that may be processed before the FM child task sleeps (gives up the CPU).

FM_CHILD_FILE_SLEEP_MS defines the length of time (in milli-secs) before the FM child task wakes (re-acquires the CPU). Note that many platforms will limit the precision of this value.

Thus the combination of the 3 values control CPU use by the FM child task. Using a smaller block size minimizes the amount of RAM used by the file I/O buffer, but at the expense of file efficiency. Adjust each of the values such that the combination is appropriate for the target platform.

For example, if the block size is 2048 and the loop count is 16 and the sleep time is 20, then while processing a 1 Mbyte file there will be 32 sleep cycles of 20ms each, for a total task delay of 0.64 seconds.

Limits:

FM_CHILD_FILE_BLOCK_SIZE: The FM application limits this value to be no less than 256 bytes and no greater than 32KB.

FM_CHILD_FILE_LOOP_COUNT: The FM application limits this value to be non-zero. There is no upper limit - a very large number effectively means that the FM child task will not surrender the CPU to other lower priority tasks.

FM_CHILD_FILE_SLEEP_MS: The FM application limits this value to be no no greater than 100 ms. The value zero generally means a very short task delay - refer to the target platform documentation for specifics.

Definition at line 228 of file fm_platform_cfg.h.

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

10.4.2.5 FM_CHILD_FILE_LOOP_COUNT

```
#define FM_CHILD_FILE_LOOP_COUNT 16
```

Definition at line 229 of file fm_platform_cfg.h.

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

10.4.2.6 FM_CHILD_FILE_SLEEP_MS

```
#define FM_CHILD_FILE_SLEEP_MS 20
```

Definition at line 230 of file fm_platform_cfg.h.

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

10.4.2.7 FM_CHILD_QUEUE_DEPTH

```
#define FM_CHILD_QUEUE_DEPTH 3
```

Child Task Command Queue Entry Count.

Description:

This definition sets the array depth for the command arguments queue in the FM main task to FM child task handshake interface. The value sets the upper limit for the number of commands that can be waiting in the queue to be processed by the low priority FM child task. A multi-entry command queue prevents the occasional slow command from being rejected because the child task has not yet completed the previous slow command.

Limits:

The FM application limits this value to be no less than 1 and no greater than 10. There must be at least one because this is the method for passing command arguments from the parent to the child task. The upper limit is arbitrary.

Definition at line 285 of file fm_platform_cfg.h.

Referenced by FM_ChildLoop(), FM_ChildProcess(), FM_InvokeChildTask(), and FM_VerifyChildTask().

10.4.2.8 FM_CHILD_SEM_NAME

```
#define FM_CHILD_SEM_NAME "FM_CHILD_SEM"
```

Child Task Semaphore Name - cFE object name.

Description:

This definition sets the FM child task semaphore object name. The semaphore object name is required during semaphore creation by cFE Executive Services.

Limits:

FM 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 object names.

Definition at line 353 of file fm_platform_cfg.h.

Referenced by FM_ChildInit().

10.4.2.9 FM_CHILD_STAT_SLEEP_FILECOUNT

```
#define FM_CHILD_STAT_SLEEP_FILECOUNT 0
```

Definition at line 266 of file fm_platform_cfg.h.

Referenced by FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), and FM_ChildSleepStat().

10.4.2.10 FM_CHILD_STAT_SLEEP_MS

```
#define FM_CHILD_STAT_SLEEP_MS 0
```

Child file stat sleep.

Description:

OS_stat is a CPU intensive call. FM uses the OS_stat call to query a file's size, date, and mode when setting up directory listings. Querying a large number of files and/or files large in size when processing directory listing commands can cause FM to hog the CPU. To mitigate this, options to sleep a configurable number of milliseconds between calls to OS_stat for a configurable number of files in a directory listing is provided. A large sleep cycle will not hang the CPU but it may take a long time for directory listing to complete. A shorter sleep cycle will speed up the directory listing commands but may cause FM to hog the CPU.

FM_CHILD_STAT_SLEEP_MS: The number of milliseconds to sleep each cycle. One cycle is FM_CHILD_STAT_S↩ LEEP_FILECOUNT.

FM_CHILD_STAT_SLEEP_FILECOUNT: The number of files to process (OS_stat) before sleeping FM_CHILD_ST↩ AT_SLEEP_MS. Works in tandem with FM_CHILD_STAT_SLEEP_MS to reduce CPU hogging while allowing slightly more customization to balance time the operator is waiting to get data back from a directory listing versus FM hogging the CPU with calls to OS_stat

In short: High SLEEP_MS means less CPU hogging by FM but a longer time to process a dir listing command Low SLEEP_MS means more potential CPU hogging by FM but shorter time to process a dir listing command High FILEC↩ OUNT means more potential CPU hogging by FM but a shorter time to process a dir listing command Low FILECOUNT means less CPU hogging by FM but longer time to process a dir listing command

Limits:

The default is zero unless the mission needs require them to be changed.

Definition at line 265 of file fm_platform_cfg.h.

Referenced by FM_ChildSleepStat().

10.4.2.11 FM_CHILD_TASK_NAME

```
#define FM_CHILD_TASK_NAME "FM_CHILD_TASK"
```

Child Task Name - cFE object name.

Description:

This definition sets the FM child task object name. The task object name is required during child task creation by cFE Executive Services.

Limits:

FM 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 object names.

Definition at line 299 of file fm_platform_cfg.h.

Referenced by FM_ChildInit().

10.4.2.12 FM_CHILD_TASK_PRIORITY

```
#define FM_CHILD_TASK_PRIORITY 205
```

Child Task Execution Priority.

Description:

This parameter sets the execution priority for the FM child task. It is highly recommended that this assignment be made by someone familiar with the system requirements for tasks running on the target platform. Note: This parameter is VxWorks® specific. Not all operating systems set task priority this way.

Limits:

Value to be no less than 1 and no greater than 255.

Priority Values:

Note that a small value has higher priority than a large value. Thus, 100 is higher priority than 150. It is also necessary to ensure that a child task has lower priority than its parent. It should be clear that a child task that runs ahead of its parent defeats the purpose of having a child task to run in the background.

Definition at line 338 of file fm_platform_cfg.h.

Referenced by FM_ChildInit().

10.4.2.13 FM_CHILD_TASK_STACK_SIZE

```
#define FM_CHILD_TASK_STACK_SIZE 20480
```

Child Task Stack Size.

Description:

This definition sets the size in bytes of the FM child task stack. It is highly recommended that this assignment be made by someone familiar with the system requirements for tasks running on the target platform.

Limits:

The FM application limits this value to be no less than 2048 and no greater than 20480. These limits are purely arbitrary and may need to be modified for specific platforms.

Definition at line 315 of file fm_platform_cfg.h.

Referenced by FM_ChildInit().

10.4.2.14 FM_DIR_LIST_FILE_DEFNAME

```
#define FM_DIR_LIST_FILE_DEFNAME "/ram/fm_dirlist.out"
```

Default Directory List Output Filename.

Description:

This definition is the default output filename used by the Get Directory List to File command handler when the output filename is not provided. The default filename is used whenever the commanded output filename is the empty string.

Limits:

The FM application does not place a limit on this configuration parameter, however the symbol must be defined and the name will be subject to the same verification tests as a commanded output filename. Set this parameter to the empty string if no default filename is desired.

Definition at line 123 of file fm_platform_cfg.h.

Referenced by FM_GetDirListFileCmd().

10.4.2.15 FM_DIR_LIST_FILE_ENTRIES

```
#define FM_DIR_LIST_FILE_ENTRIES 3000
```

Maximum Directory List Output File Entries.

Description:

This definition sets the upper limit for the number of directory entries that may be written to a Directory List output file. Directory List files are variable length, based on the number of directory entries actually written to the file. There may zero entries written to the file if the directory is empty. For most environments, this definition will play no role at all, as it will be set to a number much larger than the count of files that will ever exist in any directory at one time.

Limits:

The FM application limits this value to be no less than 100 and no greater than 10000.

Definition at line 142 of file fm_platform_cfg.h.

Referenced by FM_ChildDirListFileLoop().

10.4.2.16 FM_DIR_LIST_FILE_SUBTYPE

```
#define FM_DIR_LIST_FILE_SUBTYPE 12345
```

Directory List Output File Header Sub-Type.

Description:

This definition sets the cFE File Header sub-type value for FM Directory List data files. The value may be used to differentiate FM Directory List files from other data files.

Limits:

The FM application places no limits on this unsigned 32 bit value.

Definition at line 155 of file fm_platform_cfg.h.

Referenced by FM_ChildDirListFileInit().

10.4.2.17 FM_DIR_LIST_PKT_ENTRIES

```
#define FM_DIR_LIST_PKT_ENTRIES 20
```

Directory List Telemetry Packet Entry Count.

Description:

This definition sets the number of directory entries contained in the Directory List telemetry packet. The command handler will read directory entries until reaching the index of the start entry (set via command argument) and then continue to read directory entries and populate the telemtry packet until there are either no more unread directory entries or until the telemetry packet is full.

Limits:

The FM application limits this value to be no less than 10 and and no greater than 100. The number of directory entries in the telemetry packet will in large part determine the packet size.

Definition at line 180 of file fm_platform_cfg.h.

Referenced by FM_ChildDirListPktCmd().

10.4.2.18 FM_MISSION_REV

```
#define FM_MISSION_REV 0
```

Mission specific version number for FM 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 "fm_version.h".

Limits:

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

Definition at line 99 of file fm_platform_cfg.h.

Referenced by FM_AppInit(), and FM_NoopCmd().

10.4.2.19 FM_TABLE_CFE_NAME

```
#define FM_TABLE_CFE_NAME "FreeSpace"
```

Free Space Table Name - cFE object name.

Description:

Table object name is required during table creation.

Limits:

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

Definition at line 372 of file fm_platform_cfg.h.

Referenced by FM_TableInit().

10.4.2.20 FM_TABLE_DEF_DESC

```
#define FM_TABLE_DEF_DESC "FM File System Free Space Table"
```

Free Space Table Description.

Description:

Table files contain headers that include descriptive text. This text will be put into the file header during the table make process.

Limits:

FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Table Services for limits related to table descriptive text.

Definition at line 415 of file fm_platform_cfg.h.

10.4.2.21 FM_TABLE_DEF_NAME

```
#define FM_TABLE_DEF_NAME "/cf/fm_freespace.tbl"
```

Free Space Table Name - filename with path.

Description:

Table name with path is required to load table at startup.

Limits:

FM requires that this name be defined, but otherwise places no limits on the definition. If the named table does not exist or fails validation, the table load will fail.

Definition at line 385 of file fm_platform_cfg.h.

Referenced by FM_TableInit().

10.4.2.22 FM_TABLE_ENTRY_COUNT

```
#define FM_TABLE_ENTRY_COUNT 8
```

Number of Free Space Table Entries.

Description:

This value defines the number of entries in both the FM file system free space table and the FM file system free space telemetry packet. Note: this value does not define the number of file systems present or supported by the CFE-OSAL, the value only defines the number of file systems for which FM may be enabled to report free space data.

Limits:

FM limits this value to be not less than 1 and not greater than 32.

Definition at line 430 of file fm_platform_cfg.h.

Referenced by FM_GetFreeSpaceCmd(), FM_SetTableStateCmd(), and FM_ValidateTable().

10.4.2.23 FM_TABLE_FILENAME

```
#define FM_TABLE_FILENAME "fm_freespace.tbl"
```

Free Space Table Name - filename without path.

Description:

Table name without path defines the output name for the table file created during the table make process.

Limits:

FM requires that this name be defined, but otherwise places no limits on the definition. If the table name is not valid then the make process may fail, or the table file may be unloadable to the target hardware.

Definition at line 400 of file fm_platform_cfg.h.

10.4.2.24 FM_TABLE_VALIDATION_ERR

```
#define FM_TABLE_VALIDATION_ERR (-1)
```

Table Data Validation Error Code.

Description:

Table data is verified during the table load process. Should the validation process fail, this value will be returned by FM to cFE Table Services and displayed in an event message.

Limits:

FM requires that this value be defined, but otherwise places no limits on the definition. Refer to cFE Table Services for limits related to error return values.

Definition at line 445 of file fm_platform_cfg.h.

Referenced by FM_ValidateTable().

10.5 CFS File Manager Event IDs

Macros

- #define FM_STARTUP_EID 1

  FM Initialization Event ID.
- #define FM_STARTUP_EVENTS_ERR_EID 2

  FM Initialization Register For Event Services Failed Event ID.
- #define FM_STARTUP_CREAT_PIPE_ERR_EID 3

  FM Initialization Create SB Input Pipe Failed Event ID.
- #define FM_STARTUP_SUBSCRIB_HK_ERR_EID 4

  FM Initialization Subscribe to HK Request Failed Event ID.
- #define FM_STARTUP_SUBSCRIB_GCMD_ERR_EID 5

  FM Initialization Subscribe to FM Commands Failed Event ID.
- #define FM_STARTUP_TABLE_INIT_ERR_EID 6

  FM Initialization Register Free Space Table Failed Event ID.
- #define FM_SB_RECEIVE_ERR_EID 7

  FM Main Loop Receive from Software Bus Failed Event ID.
- #define FM_EXIT_ERR_EID 8

  FM Application Termination Event ID.
- #define FM_MID_ERR_EID 9

  FM Main Loop Message ID Invalid Event ID.
- #define FM_CC_ERR_EID 10

  FM Main Loop Command Code Invalid Event ID.
- #define FM_HK_REQ_ERR_EID 11

  FM Command Packet Length Invalid Event ID.
- #define FM_NOOP_CMD_EID 12

  FM No-op Command Event ID.
- #define FM_NOOP_PKT_ERR_EID 13

  FM No-op Command Length Invalid Event ID.
- #define FM_RESET_CMD_EID 14

  FM Reset Counters Command Event ID.
- #define FM_RESET_PKT_ERR_EID 15

  FM Reset Counters Command Length Invalid Event ID.
- #define FM_COPY_CMD_EID 16

  FM Copy File Command Event ID.
- #define FM_COPY_PKT_ERR_EID 17

  FM Copy File Command Length Invalid Event ID.
- #define FM_COPY_OVR_ERR_EID 18

  FM Copy File Command Overwrite Invalid Event ID.
- #define FM_COPY_OS_ERR_EID 19

  FM Copy File Command OS Error Event ID.
- #define FM_MOVE_CMD_EID 20

  FM Move File Command Event ID.
- #define FM_MOVE_PKT_ERR_EID 21

  FM Move File Command Length Invalid Event ID.
- #define FM_MOVE_OVR_ERR_EID 22

FM Move File Command Overwrite Invalid Event ID.

- #define FM_MOVE_OS_ERR_EID 23

  FM Move File Command OS Error Event ID.
- #define FM_RENAME_CMD_EID 24

  FM Rename File Command Event ID.
- #define FM_RENAME_PKT_ERR_EID 25

  FM Rename File Command Length Invalid Event ID.
- #define FM_RENAME_OVR_ERR_EID 26

  FM Rename File Command Overwrite Invalid Event ID.
- #define FM_RENAME_OS_ERR_EID 27

  FM Rename File Command OS Error Event ID.
- #define FM_DELETE_CMD_EID 28

  FM Delete File Command Event ID.
- #define FM_DELETE_PKT_ERR_EID 29

  FM Delete File Command Length Invalid Event ID.
- #define FM_DELETE_OS_ERR_EID 30

  FM Delete File Command OS Error Event ID.
- #define FM_DELETE_ALL_CMD_EID 31

  FM Delete All Files Command Event ID.
- #define FM_DELETE_ALL_FILES_ND_WARNING_EID 32

  FM Delete All Files Unable To Delete All Event ID.
- #define FM_DELETE_ALL_SKIP_WARNING_EID 33

  FM Delete All Files Directories Skipped Event ID.
- #define FM_DELETE_ALL_PKT_ERR_EID 34

  FM Delete All Files Command Length Invalid Event ID.
- #define FM_DELETE_ALL_OS_ERR_EID 35

  FM Delete All Files Command OS Error Event ID.
- #define FM_DECOM_CMD_EID 36

  FM Decompress File Command Event ID.
- #define FM_DECOM_PKT_ERR_EID 37

  FM Decompress File Command Length Invalid Event ID.
- #define FM_DECOM_CFE_ERR_EID 38

  FM Decompress File Decompression Failed Event ID.
- #define FM_CONCAT_CMD_EID 39

  FM Concat Files Command Event ID.
- #define FM_CONCAT_PKT_ERR_EID 40

  FM Concat Files Command Length Invalid Event ID.
- #define FM_CONCAT_OSCPY_ERR_EID 41

  FM Concat Files Copy Failed Event ID.
- #define FM_CONCAT_OPEN_SRC2_ERR_EID 42

  FM Concat Files Command Open Second Source File Failed Event ID.
- #define FM_CONCAT_OPEN_TGT_ERR_EID 43

  FM Concat Files Command Open Target File Failed Event ID.
- #define FM_CONCAT_OSRD_ERR_EID 44

  FM Concat Files Command Read Second Source File Failed Event ID.
- #define FM_CONCAT_OSWR_ERR_EID 45

  FM Concat Files Command Write Target File Failed Event ID.

- #define FM_GET_FILE_INFO_CMD_EID 46

  FM Get File Info Command Event ID.
- #define FM_GET_FILE_INFO_STATE_WARNING_EID 47

  FM Get File Info Unable To Compute CRC File State Invalid Event ID.
- #define FM_GET_FILE_INFO_TYPE_WARNING_EID 48

  FM Get File Info Unable To Compute CRC, CRC Type Invalid Event ID.
- #define FM_GET_FILE_INFO_OPEN_ERR_EID 49

  FM Get File Info Unable To Compute CRC File Open Failed Event ID.
- #define FM_GET_FILE_INFO_READ_WARNING_EID 50

  FM Get File Info Unable To Compute CRC File Read Failed Event ID.
- #define FM_GET_FILE_INFO_PKT_ERR_EID 51

  FM Get File Info Command Length Invalid Event ID.
- #define FM_GET_FILE_INFO_SRC_ERR_EID 52

  FM Get File Info Command Filename Invalid Event ID.
- #define FM_GET_OPEN_FILES_CMD_EID 53

  FM Get Open Files Command Event ID.
- #define FM_GET_OPEN_FILES_PKT_ERR_EID 54

  FM Get Open Files Command Length Invalid Event ID.
- #define FM_CREATE_DIR_CMD_EID 55

  FM Create Directory Command Event ID.
- #define FM_CREATE_DIR_PKT_ERR_EID 56

  FM Create Directory Command Length Invalid Event ID.
- #define FM_CREATE_DIR_OS_ERR_EID 57

  FM Create Directory Command OS Error Event ID.
- #define FM_DELETE_DIR_CMD_EID 58

  FM Delete Directory Command Event ID.
- #define FM_DELETE_DIR_PKT_ERR_EID 59

  FM Delete Directory Command Length Invalid Event ID.
- #define FM_DELETE_DIR_EMPTY_ERR_EID 60

  FM Delete Directory Command Failed Directory Not Empty Event ID.
- #define FM_DELETE_OPENDIR_OS_ERR_EID 61

  FM Delete Directory, Direcotry Open Failed Event ID.
- #define FM_DELETE_RMDIR_OS_ERR_EID 62

  FM Delete Directory Remove Directory Failed Event ID.
- #define FM_GET_DIR_FILE_CMD_EID 63

  FM Directory List To File Command Event ID.
- #define FM_GET_DIR_FILE_PKT_ERR_EID 64

  FM Directory List To File Command Length Invalid Event ID.
- #define FM_GET_DIR_FILE_WARNING_EID 65

  FM Directory List To File Command Combined Path and Name Too Long Event ID.
- #define FM_GET_DIR_FILE_OSOPENDIR_ERR_EID 66

  FM Directory List To File Directory Open Failed Event ID.
- #define FM_GET_DIR_FILE_WRBLANK_ERR_EID 67

  FM Directory List To File Write Blank Stats Failed Event ID.
- #define FM_GET_DIR_FILE_WRHDR_ERR_EID 68

  FM Directory List To File Write Header Failed Event ID.
- #define FM_GET_DIR_FILE_OSCREAT_ERR_EID 69

FM Directory List To File Create File Failed Event ID.

- #define FM_GET_DIR_FILE_WRENTRY_ERR_EID 70

  FM Directory List To File Write Entry Failed Event ID.
- #define FM_GET_DIR_FILE_UPSTATS_ERR_EID 71

  FM Directory List To File Write Update Stats Failed Event ID.
- #define FM_GET_DIR_PKT_CMD_EID 72

  FM Directory List To Packet Command Event ID.
- #define FM_GET_DIR_PKT_WARNING_EID 73

  FM Directory List To Packet Command Directory and Entry Too Long Event ID.
- #define FM_GET_DIR_PKT_PKT_ERR_EID 74

  FM Directory List To Packet Command Length Invalid Event ID.
- #define FM_GET_DIR_PKT_OS_ERR_EID 75

  FM Directory List To Packet Directory Open Failed Event ID.
- #define FM_GET_FREE_SPACE_CMD_EID 76

  FM Get Free Space Command Event ID.
- #define FM_GET_FREE_SPACE_PKT_ERR_EID 77

  FM Get Free Space Command Length Invalid Event ID.
- #define FM_GET_FREE_SPACE_TBL_ERR_EID 78

  FM Get Free Space Table Not Loaded Event ID.
- #define FM_SET_TABLE_STATE_CMD_EID 79

  FM Set Table State Command Event ID.
- #define FM_SET_TABLE_STATE_PKT_ERR_EID 80

  FM Set Table State Command Length Invalid Event ID.
- #define FM_SET_TABLE_STATE_TBL_ERR_EID 81

  FM Set Table State Command Table Not Loaded Event ID.
- #define FM_SET_TABLE_STATE_ARG_IDX_ERR_EID 82

  FM Set Table State Command Index Invalid Event ID.
- #define FM_SET_TABLE_STATE_ARG_STATE_ERR_EID 83

  FM Set Table State Command State Invalid Event ID.
- #define FM_SET_TABLE_STATE_UNUSED_ERR_EID 84

  FM Set Table State Command Unused Entry Event ID.
- #define FM_TABLE_VERIFY_EMPTY_ERR_EID 85

  FM Free Space Table Verification Failed Empty Name Event ID.
- #define FM_TABLE_VERIFY_TOOLONG_ERR_EID 86

  FM Free Space Table Verification Failed Name Too Long Event ID.
- #define FM_TABLE_VERIFY_BAD_STATE_ERR_EID 88

  FM Free Space Table Verification Failed State Invalid Event ID.
- #define FM_CHILD_INIT_EID 89

  FM Child Task Initialization Complete Event ID.
- #define FM_CHILD_INIT_SEM_ERR_EID 90

  FM Child Task Initialization Create Semaphore Failed Event ID.
- #define FM_CHILD_INIT_QSEM_ERR_EID 91

  FM Child Task Initialization Create Queue Count Semaphore Failed Event ID.
- #define FM_CHILD_INIT_CREATE_ERR_EID 92

  FM Child Task Initialization Create Task Failed Event ID.
- #define FM_CHILD_TERM_EMPTYQ_ERR_EID 93

  FM Child Task Termination Error Empty Queue Event ID.

- #define FM_CHILD_TERM_QIDX_ERR_EID 94

  FM Child Task Termination Error Invalid Queue Index Event ID.
- #define FM_CHILD_TERM_SEM_ERR_EID 95

  FM Child Task Termination Error Semaphore Take Failed Event ID.
- #define FM_CHILD_EXE_ERR_EID 96

  FM Child Task Command Code Invalid Event ID.
- #define FM_TABLE_VERIFY_EID 97

  FM Free Space Table Validation Results Event ID.
- #define FM_SET_PERM_ERR_EID 98

  FM Set Permissions Command Length Invalid Event ID.
- #define FM_SET_PERM_CMD_EID 99

  FM Set Permissions Command Event ID.
- #define FM_SET_PERM_OS_ERR_EID 100

  FM Set Permissions Command Chmod Error Event ID.
- #define FM_TABLE_VERIFY_NULL_PTR_ERR_EID 101

  FM Free Space Table Verification Failed Null Pointer Detected.
- #define FM_SB_RECEIVE_NULL_PTR_ERR_EID 102

  FM Main Loop Software Bus Returned NULL On Success Event ID.
- #define FM_OS_SYS_STAT_ERR_EID 103

  FM Get Free Space Get File SYstem Stats Failed Event ID.
- #define FM_FNAME_INVALID_EID_OFFSET 0
- #define FM_FNAME_DNE_EID_OFFSET 1
- #define FM_FNAME_EXIST_EID_OFFSET 1 /∗ mutually exclusive with DNE ∗/
- #define FM_FNAME_ISDIR_EID_OFFSET 2
- #define FM_FNAME_ISFILE_EID_OFFSET 2 /∗ mutually exclusive with ISDIR ∗/
- #define FM_FNAME_ISOPEN_EID_OFFSET 3
- #define FM_FNAME_ISCLOSED_EID_OFFSET 4
- #define FM_FNAME_UNKNOWN_EID_OFFSET 5
- #define FM_FNAME_NUM_OFFSETS 6
- #define FM_CHILD_DISABLED_EID_OFFSET 0
- #define FM_CHILD_Q_FULL_EID_OFFSET 1
- #define FM_CHILD_BROKEN_EID_OFFSET 2
- #define FM_CHILD_NUM_OFFSETS 3
- #define FM_COPY_SRC_BASE_EID 151

  FM Child Task Copy File Source Filename Error Base ID.
- #define FM_COPY_SRC_INVALID_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Copy File Source Name Invalid Event ID.
- #define FM_COPY_SRC_DNE_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Copy File Source File Does Not Exist Event ID.
- #define FM_COPY_SRC_ISDIR_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Copy File Source File Name Is Directory Event ID.
- #define FM_COPY_SRC_UNKNOWN_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_UNKNOWN_↩EID_OFFSET)

  FM Child Task Copy File Source File Name Has Unknown State Event ID.
- #define FM_COPY_TGT_BASE_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Copy File Target Filename Error Base ID.

- #define FM_COPY_TGT_INVALID_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Copy File Target Filename Invalid Event ID.
- #define FM_COPY_TGT_EXIST_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)

  FM Child Task Copy File Target File Already Exists Event ID.
- #define FM_COPY_TGT_ISDIR_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Copy File Target Filename Is A Directory Event ID.
- #define FM_COPY_TGT_ISOPEN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_OF↩FSET)

  FM Child Task Copy File Target Filename Exists As Open File Event ID.
- #define FM_COPY_TGT_UNKNOWN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_UNKNOWN_E↩ID_OFFSET)

  FM Child Task Copy File Target Filename Unknown State Event ID.
- #define FM_COPY_CHILD_BASE_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Copy File Child Task Error Base ID.
- #define FM_COPY_CHILD_DISABLED_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_DISABLED↩_EID_OFFSET)

  FM Child Task Copy File Child Task Disabled Event ID.
- #define FM_COPY_CHILD_FULL_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_O↩FFSET)

  FM Child Task Copy File Child Task Queue Full Event ID.
- #define FM_COPY_CHILD_BROKEN_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_BROKEN_EI↩D_OFFSET)

  FM Child Task Copy File Child Task Interface Broken Event ID.
- #define FM_MOVE_SRC_BASE_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Move File Source Filename Error Base ID.
- #define FM_MOVE_SRC_INVALID_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Move File Source Filename Invalid Event ID.
- #define FM_MOVE_SRC_DNE_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Move File Source File Does Not Exist Event ID.
- #define FM_MOVE_SRC_ISDIR_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Move File Source Filename Is A Directory Event ID.
- #define FM_MOVE_TGT_BASE_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Move File Target Filename Error Base ID.
- #define FM_MOVE_TGT_INVALID_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Move File Target Filename Invalid Event ID.
- #define FM_MOVE_TGT_EXIST_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFS↩ET)

  FM Child Task Move File Target File Already Exists Event ID.
- #define FM_MOVE_TGT_ISDIR_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Move File Target Filename Is A Directory Event ID.
- #define FM_MOVE_TGT_ISOPEN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_O↩FFSET)

  FM Child Task Move File Target File Exists As An Open File Event ID.
- #define FM_MOVE_TGT_UNKNOWN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_UNKNOWN_↩EID_OFFSET)

FM Child Task Move File Target File Has Unknown State Event ID.

- #define FM_MOVE_CHILD_BASE_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Move File Child Task Error Base ID.

- #define FM_MOVE_CHILD_DISABLED_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_DISABLED↩_EID_OFFSET)

  FM Child Task Move File Child Task Disabled Event ID.

- #define FM_MOVE_CHILD_FULL_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_O↩FFSET)

  FM Child Task Move File Child Task Queue Full Event ID.

- #define FM_MOVE_CHILD_BROKEN_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_BROKEN_EI↩D_OFFSET)

  FM Child Task Move File Child Task Inteface Broken Event ID.

- #define FM_RENAME_SRC_BASE_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Rename File Source Filename Error Base ID.

- #define FM_RENAME_SRC_INVALID_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_INVALID_↩EID_OFFSET)

  FM Child Task Rename File Source Filename Invalid Event ID.

- #define FM_RENAME_SRC_DNE_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_DNE_EID_OF↩FSET)

  FM Child Task Rename File Source File Does Not Exist Event ID.

- #define FM_RENAME_SRC_ISDIR_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_ISDIR_EID_↩OFFSET)

  FM Child Task Rename File Source Filename Is Directory Event ID.

- #define FM_RENAME_TGT_BASE_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Rename File Target Filename Error Base ID.

- #define FM_RENAME_TGT_INVALID_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_INVALID_↩EID_OFFSET)

  FM Child Task Rename File Target Filename Invalid Event ID.

- #define FM_RENAME_TGT_EXIST_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_EXIST_EID_↩OFFSET)

  FM Child Task Rename File Target File Already Exists Event ID.

- #define FM_RENAME_TGT_ISDIR_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISDIR_EID_O↩FFSET)

  FM Child Task Rename File Target Filename Is Directory Event ID.

- #define FM_RENAME_TGT_ISOPEN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISOPEN_E↩ID_OFFSET)

  FM Child Task Rename File Target Filename Exists As Open File Event ID.

- #define FM_RENAME_TGT_UNKNOWN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_UNKNO↩WN_EID_OFFSET)

  FM Child Task Rename File Target Filename Has Unknown State Event ID.

- #define FM_RENAME_CHILD_BASE_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Rename File Child Task Error Base ID.

- #define FM_RENAME_CHILD_DISABLED_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_DISA↩BLED_EID_OFFSET)

  FM Child Task Rename File Child Task Disabled Event ID.

- #define FM_RENAME_CHILD_FULL_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_Q_FULL_↩EID_OFFSET)

  FM Child Task Rename File Child Task Queue Full Event ID.

- #define FM_RENAME_CHILD_BROKEN_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Rename File Child Task Inteface Broken Event ID.
- #define FM_DELETE_SRC_BASE_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete File Source Filename Error Base ID.
- #define FM_DELETE_SRC_INVALID_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Delete File Source Filename Invalid Event ID.
- #define FM_DELETE_SRC_DNE_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Delete File Source File Does Not Exist Event ID.
- #define FM_DELETE_SRC_ISDIR_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Delete File Source Filename Is Directory Event ID.
- #define FM_DELETE_SRC_OPEN_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)

  FM Child Task Delete File File Is Open Event ID.
- #define FM_DELETE_CHILD_BASE_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Delete File Child Task Error Base ID.
- #define FM_DELETE_CHILD_DISABLED_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Delete File Child Task Disabled Event ID.
- #define FM_DELETE_CHILD_FULL_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Delete File Child Task Queue Full Event ID.
- #define FM_DELETE_CHILD_BROKEN_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Delete File Child Task Inteface Broken Event ID.
- #define FM_DELETE_ALL_SRC_BASE_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete All Files Directory Error Base ID.
- #define FM_DELETE_ALL_SRC_INVALID_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Delete All Files Directory Name Invalid Event ID.
- #define FM_DELETE_ALL_SRC_DNE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Delete All Files Directory Does Not Exist Event ID.
- #define FM_DELETE_ALL_SRC_FILE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_ISFILE_EID_OFFSET)

  FM Child Task Delete All Files Directory Name Is A File Event ID.
- #define FM_DELETE_ALL_CHILD_BASE_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Delete All Files Child Task Error Base ID.
- #define FM_DELETE_ALL_CHILD_DISABLED_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Delete All Files Child Task Disabled Event ID.
- #define FM_DELETE_ALL_CHILD_FULL_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Delete All Files Child Task Queue Full Event ID.

- #define FM_DELETE_ALL_CHILD_BROKEN_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Delete All Files Child Task Inteface Broken Event ID.
- #define FM_DECOM_SRC_BASE_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Decompress File Source Filename Error Base ID.
- #define FM_DECOM_SRC_INVALID_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Decompress File Source Filename Invalid Event ID.
- #define FM_DECOM_SRC_DNE_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Decompress File Source Filename Does Not Exist Event ID.
- #define FM_DECOM_SRC_ISDIR_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Decompress File Source Filename Is A Directory Event ID.
- #define FM_DECOM_SRC_OPEN_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)

  FM Child Task Decompress File Source File Is Open Event ID.
- #define FM_DECOM_TGT_BASE_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Decompress File Target Filename Error Base ID.
- #define FM_DECOM_TGT_INVALID_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Decompress File Target Filename Invalid Event ID.
- #define FM_DECOM_TGT_EXIST_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)

  FM Child Task Decompress File Target File Already Exists Event ID.
- #define FM_DECOM_TGT_ISDIR_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Decompress File Target Filename Is Directory Event ID.
- #define FM_DECOM_CHILD_BASE_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Decompress File Child Task Error Base ID.
- #define FM_DECOM_CHILD_DISABLED_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Decompress File Child Task Disabled Event ID.
- #define FM_DECOM_CHILD_FULL_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Decompress File Child Task Queue Full Event ID.
- #define FM_DECOM_CHILD_BROKEN_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Decompress File Child Task Inteface Broken Event ID.
- #define FM_CONCAT_SRC1_BASE_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Concat Files Source 1 Filename Error Base ID.
- #define FM_CONCAT_SRC1_INVALID_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Concat Files Source 1 Filename Invalid Event ID.
- #define FM_CONCAT_SRC1_DNE_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Concat Files Source 1 File Does Not Exist Event ID.
- #define FM_CONCAT_SRC1_ISDIR_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

FM Child Task Concat Files Source 1 Filename Is Directory Event ID.

- #define FM_CONCAT_SRC1_OPEN_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISOPEN_↩EID_OFFSET)

  FM Child Task Concat Files Source 1 File Already Open Event ID.

- #define FM_CONCAT_SRC2_BASE_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Source 2 Filename Error Base ID.

- #define FM_CONCAT_SRC2_INVALID_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_INVALI↩D_EID_OFFSET)

  FM Child Task Concat Files Source 2 Filename Invalid Event ID.

- #define FM_CONCAT_SRC2_DNE_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_DNE_EID_↩OFFSET)

  FM Child Task Concat Files Source 2 File Does Not Exist Event ID.

- #define FM_CONCAT_SRC2_ISDIR_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISDIR_EID↩_OFFSET)

  FM Child Task Concat Files Source 2 Filename Is Directory Event ID.

- #define FM_CONCAT_SRC2_OPEN_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISOPEN_↩EID_OFFSET)

  FM Child Task Concat Files Source 2 File Already Open Event ID.

- #define FM_CONCAT_TGT_BASE_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Target Filename Error Base ID.

- #define FM_CONCAT_TGT_INVALID_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_INVALID_E↩ID_OFFSET)

  FM Child Task Concat Files Target Filename Invalid Event ID.

- #define FM_CONCAT_TGT_EXIST_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_EXIST_EID_↩OFFSET)

  FM Child Task Concat Files Target Filename Already Exists Event ID.

- #define FM_CONCAT_TGT_ISDIR_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_ISDIR_EID_O↩FFSET)

  FM Child Task Concat Files Target Filename Is Directory Event ID.

- #define FM_CONCAT_CHILD_BASE_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Child Task Error Base ID.

- #define FM_CONCAT_CHILD_DISABLED_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_DISA↩BLED_EID_OFFSET)

  FM Child Task Concat Files Child Task Disabled Event ID.

- #define FM_CONCAT_CHILD_FULL_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_Q_FULL_E↩ID_OFFSET)

  FM Child Task Concat Files Child Task Queue Full Event ID.

- #define FM_CONCAT_CHILD_BROKEN_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_BROK↩EN_EID_OFFSET)

  FM Child Task Concat Files Child Task Inteface Broken Event ID.

- #define FM_FILE_INFO_CHILD_BASE_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_NUM_OFFSE↩TS)

  FM Child Task Get File Info Child Task Error Base ID.

- #define FM_FILE_INFO_CHILD_DISABLED_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_D↩ISABLED_EID_OFFSET)

  FM Child Task Get File Info Child Task Disabled Event ID.

- #define FM_FILE_INFO_CHILD_FULL_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_Q_FUL↩L_EID_OFFSET)

  FM Child Task Get File Info Child Task Queue Full Event ID.

- #define FM_FILE_INFO_CHILD_BROKEN_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Get File Info Child Task Inteface Broken Event ID.
- #define FM_CREATE_DIR_SRC_BASE_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Create Directory, Directory Error Base ID.
- #define FM_CREATE_DIR_SRC_INVALID_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Create Directory Name Invalid Event ID.
- #define FM_CREATE_DIR_SRC_DNE_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Create Directory Name Exists As File Event ID.
- #define FM_CREATE_DIR_SRC_ISDIR_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Create Directory Already Exists Event ID.
- #define FM_CREATE_DIR_CHILD_BASE_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Create Directory Child Task Error Base ID.
- #define FM_CREATE_DIR_CHILD_DISABLED_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Create Directory Child Task Disabled Event ID.
- #define FM_CREATE_DIR_CHILD_FULL_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Create Directory Child Task Queue Full Event ID.
- #define FM_CREATE_DIR_CHILD_BROKEN_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Create Directory Child Task Inteface Broken Event ID.
- #define FM_DELETE_DIR_SRC_BASE_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete Directory, Directory Error Base ID.
- #define FM_DELETE_DIR_SRC_INVALID_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Delete Directory Name Invalid Event ID.
- #define FM_DELETE_DIR_SRC_DNE_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Delete Directory Name Exists As File Event ID.
- #define FM_DELETE_DIR_CHILD_BASE_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Delete Directory Child Task Error Base ID.
- #define FM_DELETE_DIR_CHILD_DISABLED_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Delete Directory Child Task Disabled Event ID.
- #define FM_DELETE_DIR_CHILD_FULL_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Delete Directory Child Task Queue Full Event ID.
- #define FM_DELETE_DIR_CHILD_BROKEN_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Delete Directory Child Task Inteface Broken Event ID.

- #define FM_GET_DIR_FILE_SRC_BASE_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_O↩FFSETS)

  FM Child Task Directory List to File Source Filename Error Base ID.
- #define FM_GET_DIR_FILE_SRC_INVALID_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAM↩E_INVALID_EID_OFFSET)

  FM Child Task Directory List to File Directory Name Invalid Event ID.
- #define FM_GET_DIR_FILE_SRC_DNE_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_D↩NE_EID_OFFSET)

  FM Child Task Directory List to File Directory Does Not Exist Event ID.
- #define FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_I↩SFILE_EID_OFFSET)

  FM Child Task Directory List to File Directory Name Is File Event ID.
- #define FM_GET_DIR_FILE_TGT_BASE_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_NUM_↩OFFSETS)

  FM Child Task Directory List to File Target Error Base ID.
- #define FM_GET_DIR_FILE_TGT_INVALID_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME↩_INVALID_EID_OFFSET)

  FM Child Task Directory List to File Target Filename Invalid Event ID.
- #define FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_I↩SDIR_EID_OFFSET)

  FM Child Task Directory List to File Target Filename Is Directory Event ID.
- #define FM_GET_DIR_FILE_CHILD_BASE_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_NUM↩_OFFSETS)

  FM Child Task Directory List to File Child Task Error Base ID.
- #define FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_↩CHILD_DISABLED_EID_OFFSET)

  FM Child Task Directory List to File Child Task Disabled Event ID.
- #define FM_GET_DIR_FILE_CHILD_FULL_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD↩_Q_FULL_EID_OFFSET)

  FM Child Task Directory List to File Child Task Queue Full Event ID.
- #define FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_C↩HILD_BROKEN_EID_OFFSET)

  FM Child Task Directory List to File Child Task Inteface Broken Event ID.
- #define FM_GET_DIR_PKT_SRC_BASE_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_NUM_↩OFFSETS)

  FM Child Task Directory List to Packet Directory Error Base ID.
- #define FM_GET_DIR_PKT_SRC_INVALID_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME↩_INVALID_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Name Invalid Event ID.
- #define FM_GET_DIR_PKT_SRC_DNE_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_DN↩E_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Does Not Exist Event ID.
- #define FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_I↩SFILE_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Is File Event ID.
- #define FM_GET_DIR_PKT_CHILD_BASE_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_NUM↩_OFFSETS)

  FM Child Task Directory List to Packet Child Task Error Base ID.

- #define FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_C↩HILD_DISABLED_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Disabled Event ID.
- #define FM_GET_DIR_PKT_CHILD_FULL_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CHILD↩_Q_FULL_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Queue Full Event ID.
- #define FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CH↩ILD_BROKEN_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Inteface Broken Event ID.

10.5.1 Detailed Description

10.5.2 Macro Definition Documentation

10.5.2.1 FM_CC_ERR_EID

```
#define FM_CC_ERR_EID 10
```

FM Main Loop Command Code Invalid Event ID.

Type: ERROR

Cause

This event message is issued when the File Manager application has received a command packet with an unexpected command code value.

Mal-formed command packets are generally prevented by the ground system. Therefore, the source for the problem command is likely to be one of the on-board tables that contain commands.

Definition at line 179 of file fm_events.h.

Referenced by FM_ProcessCmd().

10.5.2.2 FM_CHILD_BROKEN_EID_OFFSET

```
#define FM_CHILD_BROKEN_EID_OFFSET 2
```

Definition at line 1531 of file fm_events.h.

Referenced by FM_VerifyChildTask().

10.5.2.3 FM_CHILD_DISABLED_EID_OFFSET

```
#define FM_CHILD_DISABLED_EID_OFFSET 0
```

Definition at line 1529 of file fm_events.h.

Referenced by FM_VerifyChildTask().

10.5.2.4 FM_CHILD_EXE_ERR_EID

```
#define FM_CHILD_EXE_ERR_EID 96
```

FM Child Task Command Code Invalid Event ID.

Type: ERROR

Cause

This event message indicates that the FM child task is unable to process the current handshake request. Either the handshake queue index or the handshake command code is invalid. This error suggests that either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface data. It may be necessary to restart the FM application to resync the handshake interface.

Definition at line 1425 of file fm_events.h.

Referenced by FM_ChildProcess().

10.5.2.5 FM_CHILD_INIT_CREATE_ERR_EID

```
#define FM_CHILD_INIT_CREATE_ERR_EID 92
```

FM Child Task Initialization Create Task Failed Event ID.

Type: ERROR

Cause

This event message indicates an unsuccessful attempt to create the low priority FM child task. Commands which would have otherwise been handed off to the child task for execution, will now be processed by the main FM application.

Definition at line 1370 of file fm_events.h.

Referenced by FM_ChildInit().

10.5.2.6 FM_CHILD_INIT_EID

```
#define FM_CHILD_INIT_EID 89
```

FM Child Task Initialization Complete Event ID.

Type: INFORMATION

Cause

This event message signals the successful completion of the initialization process for the FM child task.

Definition at line 1330 of file fm_events.h.

Referenced by FM_ChildTask().

10.5.2.7 FM_CHILD_INIT_QSEM_ERR_EID

```
#define FM_CHILD_INIT_QSEM_ERR_EID 91
```

FM Child Task Initialization Create Queue Count Semaphore Failed Event ID.

Type: ERROR

Cause

This event message indicates an unsuccessful attempt to create the queue count semphore for the FM child task. Commands which would have otherwise been handed off to the child task for execution, will now be processed by the main FM application.

Definition at line 1357 of file fm_events.h.

Referenced by FM_ChildInit().

10.5.2.8 FM_CHILD_INIT_SEM_ERR_EID

```
#define FM_CHILD_INIT_SEM_ERR_EID 90
```

FM Child Task Initialization Create Semaphore Failed Event ID.

Type: ERROR

Cause

This event message indicates an unsuccessful attempt to create the sempahore for the low priority FM child task. Commands which would have otherwise been handed off to the child task for execution, will now be processed by the main FM application.

Definition at line 1344 of file fm_events.h.

Referenced by FM_ChildInit().

10.5.2.9 FM_CHILD_NUM_OFFSETS

```
#define FM_CHILD_NUM_OFFSETS 3
```

Definition at line 1532 of file fm_events.h.

10.5.2.10 FM_CHILD_Q_FULL_EID_OFFSET

```
#define FM_CHILD_Q_FULL_EID_OFFSET 1
```

Definition at line 1530 of file fm_events.h.

Referenced by FM_VerifyChildTask().

10.5.2.11 FM_CHILD_TERM_EMPTYQ_ERR_EID

```
#define FM_CHILD_TERM_EMPTYQ_ERR_EID 93
```

FM Child Task Termination Error Empty Queue Event ID.

Type: ERROR

Cause

This event message indicates that the FM child task has suffered a fatal error and has terminated. The error occurred because the child queue was empty, indicating that the handshake between the main task and child task was broken.

Definition at line 1384 of file fm_events.h.

Referenced by FM_ChildLoop().

10.5.2.12 FM_CHILD_TERM_QIDX_ERR_EID

```
#define FM_CHILD_TERM_QIDX_ERR_EID 94
```

FM Child Task Termination Error Invalid Queue Index Event ID.

Type: ERROR

Cause

This event message indicates that the FM child task has suffered a fatal error and has terminated. The error occurred because the child read index was invalid (larger than the child queue depth).

Definition at line 1397 of file fm_events.h.

Referenced by FM_ChildLoop().

10.5.2.13 FM_CHILD_TERM_SEM_ERR_EID

```
#define FM_CHILD_TERM_SEM_ERR_EID 95
```

FM Child Task Termination Error Semaphore Take Failed Event ID.

Type: ERROR

Cause

This event message indicates that the FM child task has suffered a fatal error and has terminated. The error occurred when trying to take the child handshake semaphore.

Definition at line 1410 of file fm_events.h.

Referenced by FM_ChildLoop().

10.5.2.14 FM_CONCAT_CHILD_BASE_EID

```
#define FM_CONCAT_CHILD_BASE_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Concat Files Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 247

Definition at line 2857 of file fm_events.h.

Referenced by FM_ConcatFilesCmd().

10.5.2.15 FM_CONCAT_CHILD_BROKEN_ERR_EID

```
#define FM_CONCAT_CHILD_BROKEN_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Concat Files Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 249

Definition at line 2913 of file fm_events.h.

10.5.2.16 FM_CONCAT_CHILD_DISABLED_ERR_EID

```
#define FM_CONCAT_CHILD_DISABLED_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Concat Files Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 247

Definition at line 2871 of file fm_events.h.

10.5.2.17 FM_CONCAT_CHILD_FULL_ERR_EID

```
#define FM_CONCAT_CHILD_FULL_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Concat Files Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 248

Definition at line 2892 of file fm_events.h.

10.5.2.18 FM_CONCAT_CMD_EID

```
#define FM_CONCAT_CMD_EID 39
```

FM Concat Files Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Concat command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 577 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.19 FM_CONCAT_OPEN_SRC2_ERR_EID

```
#define FM_CONCAT_OPEN_SRC2_ERR_EID 42
```

FM Concat Files Command Open Second Source File Failed Event ID.

Type: ERROR

Cause

This event message is generated when the second source file cannot be opened.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 625 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.20 FM_CONCAT_OPEN_TGT_ERR_EID

```
#define FM_CONCAT_OPEN_TGT_ERR_EID 43
```

FM Concat Files Command Open Target File Failed Event ID.

Type: ERROR

Cause

This event message is generated when the target file cannot be opened.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 643 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.21 FM_CONCAT_OSCPY_ERR_EID

```
#define FM_CONCAT_OSCPY_ERR_EID 41
```

FM Concat Files Copy Failed Event ID.

Type: ERROR

Cause

This event message is generated when the first source file cannot be copied.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 607 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.22 FM_CONCAT_OSRD_ERR_EID

```
#define FM_CONCAT_OSRD_ERR_EID 44
```

FM Concat Files Command Read Second Source File Failed Event ID.

Type: ERROR

Cause

This event message is generated when the second source file cannot be read.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 660 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.23 FM_CONCAT_OSWR_ERR_EID

```
#define FM_CONCAT_OSWR_ERR_EID 45
```

FM Concat Files Command Write Target File Failed Event ID.

Type: ERROR

Cause

This event message is generated when the target file cannot be written.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 677 of file fm_events.h.

Referenced by FM_ChildConcatCmd().

10.5.2.24 FM_CONCAT_PKT_ERR_EID

```
#define FM_CONCAT_PKT_ERR_EID 40
```

FM Concat Files Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with an invalid length.

Definition at line 589 of file fm_events.h.

Referenced by FM_ConcatFilesCmd().

10.5.2.25 FM_CONCAT_SRC1_BASE_EID

```
#define FM_CONCAT_SRC1_BASE_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Concat Files Source 1 Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source 1 filename that is unusable for one of several reasons.

Value: 229

Definition at line 2659 of file fm_events.h.

Referenced by FM_ConcatFilesCmd().

10.5.2.26 FM_CONCAT_SRC1_DNE_ERR_EID

```
#define FM_CONCAT_SRC1_DNE_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Concat Files Source 1 File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source 1 filename that does not exist.

Value: 230

Definition at line 2687 of file fm_events.h.

10.5.2.27 FM_CONCAT_SRC1_INVALID_ERR_EID

```
#define FM_CONCAT_SRC1_INVALID_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Concat Files Source 1 Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with an invalid source 1 filename.

Value: 229

Definition at line 2673 of file fm_events.h.

10.5.2.28 FM_CONCAT_SRC1_ISDIR_ERR_EID

```
#define FM_CONCAT_SRC1_ISDIR_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Concat Files Source 1 Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source filename that is a directory.

Value: 231

Definition at line 2701 of file fm_events.h.

10.5.2.29 FM_CONCAT_SRC1_OPEN_ERR_EID

```
#define FM_CONCAT_SRC1_OPEN_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Concat Files Source 1 File Already Open Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source filename that is already open.

Value: 232

Definition at line 2715 of file fm_events.h.

10.5.2.30 FM_CONCAT_SRC2_BASE_EID

```
#define FM_CONCAT_SRC2_BASE_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Concat Files Source 2 Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source 2 filename that is unusable for one of several reasons.

Value: 235

Definition at line 2730 of file fm_events.h.

Referenced by FM_ConcatFilesCmd().

10.5.2.31 FM_CONCAT_SRC2_DNE_ERR_EID

```
#define FM_CONCAT_SRC2_DNE_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Concat Files Source 2 File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source 2 filename that does not exist.

Value: 236

Definition at line 2758 of file fm_events.h.

10.5.2.32 FM_CONCAT_SRC2_INVALID_ERR_EID

```
#define FM_CONCAT_SRC2_INVALID_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Concat Files Source 2 Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with an invalid source 2 filename.

Value: 235

Definition at line 2744 of file fm_events.h.

10.5.2.33 FM_CONCAT_SRC2_ISDIR_ERR_EID

```
#define FM_CONCAT_SRC2_ISDIR_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Concat Files Source 2 Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source filename that is a directory.

Value: 237

Definition at line 2772 of file fm_events.h.

10.5.2.34 FM_CONCAT_SRC2_OPEN_ERR_EID

```
#define FM_CONCAT_SRC2_OPEN_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Concat Files Source 2 File Already Open Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a source filename that is already open.

Value: 238

Definition at line 2786 of file fm_events.h.

10.5.2.35 FM_CONCAT_TGT_BASE_EID

```
#define FM_CONCAT_TGT_BASE_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Concat Files Target Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a target filename that is unusable for one of several reasons.

Value: 241

Definition at line 2801 of file fm_events.h.

Referenced by FM_ConcatFilesCmd().

10.5.2.36 FM_CONCAT_TGT_EXIST_ERR_EID

```
#define FM_CONCAT_TGT_EXIST_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)
```

FM Child Task Concat Files Target Filename Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a target filename that already exists.

Value: 242

Definition at line 2829 of file fm_events.h.

10.5.2.37 FM_CONCAT_TGT_INVALID_ERR_EID

```
#define FM_CONCAT_TGT_INVALID_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Concat Files Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with an invalid target filename.

Value: 241

Definition at line 2815 of file fm_events.h.

10.5.2.38 FM_CONCAT_TGT_ISDIR_ERR_EID

```
#define FM_CONCAT_TGT_ISDIR_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Concat Files Target Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Concat command packet with a target filename that is a directory.

Value: 243

Definition at line 2843 of file fm_events.h.

10.5.2.39 FM_COPY_CHILD_BASE_EID

```
#define FM_COPY_CHILD_BASE_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Copy File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 163

Definition at line 1698 of file fm_events.h.

Referenced by FM_CopyFileCmd().

10.5.2.40 FM_COPY_CHILD_BROKEN_ERR_EID

```
#define FM_COPY_CHILD_BROKEN_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Copy File Child Task Interface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 165

Definition at line 1754 of file fm_events.h.

10.5.2.41 FM_COPY_CHILD_DISABLED_ERR_EID

```
#define FM_COPY_CHILD_DISABLED_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Copy File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 163

Definition at line 1712 of file fm_events.h.

10.5.2.42 FM_COPY_CHILD_FULL_ERR_EID

```
#define FM_COPY_CHILD_FULL_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Copy File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 164

Definition at line 1733 of file fm_events.h.

10.5.2.43 FM_COPY_CMD_EID

```
#define FM_COPY_CMD_EID 16
```

FM Copy File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Copy command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 259 of file fm_events.h.

Referenced by FM_ChildCopyCmd().

10.5.2.44 FM_COPY_OS_ERR_EID

```
#define FM_COPY_OS_ERR_EID 19
```

FM Copy File Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. Verify that the target filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return value.

Definition at line 301 of file fm_events.h.

Referenced by FM_ChildCopyCmd().

10.5.2.45 FM_COPY_OVR_ERR_EID

```
#define FM_COPY_OVR_ERR_EID 18
```

FM Copy File Command Overwrite Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with an invalid overwrite argument. Overwrite must be set to TRUE (one) or FALSE (zero).

Definition at line 284 of file fm_events.h.

Referenced by FM_CopyFileCmd().

10.5.2.46 FM_COPY_PKT_ERR_EID

```
#define FM_COPY_PKT_ERR_EID 17
```

FM Copy File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with an invalid length.

Definition at line 271 of file fm_events.h.

Referenced by FM_CopyFileCmd().

10.5.2.47 FM_COPY_SRC_BASE_EID

```
#define FM_COPY_SRC_BASE_EID 151
```

FM Child Task Copy File Source Filename Error Base ID.

Type: ERROR

Cause

This is the base for a number of error events generated when the /FM_Copy is received with an unusable source filename.

Definition at line 1544 of file fm_events.h.

Referenced by FM_CopyFileCmd().

10.5.2.48 FM_COPY_SRC_DNE_ERR_EID

```
#define FM_COPY_SRC_DNE_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Copy File Source File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a source filename that does not exist.

Value: 152

Definition at line 1572 of file fm_events.h.

10.5.2.49 FM_COPY_SRC_INVALID_ERR_EID

```
#define FM_COPY_SRC_INVALID_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Copy File Source Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with an invalid source filename.

Value: 151

Definition at line 1558 of file fm_events.h.

10.5.2.50 FM_COPY_SRC_ISDIR_ERR_EID

```
#define FM_COPY_SRC_ISDIR_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Copy File Source File Name Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a source filename that is a directory.

Value: 153

Definition at line 1586 of file fm_events.h.

10.5.2.51 FM_COPY_SRC_UNKNOWN_ERR_EID

```
#define FM_COPY_SRC_UNKNOWN_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_UNKNOWN_EID_OFFSET)
```

FM Child Task Copy File Source File Name Has Unknown State Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a source filename of unknown state.

Value: 156

Definition at line 1600 of file fm_events.h.

10.5.2.52 FM_COPY_TGT_BASE_EID

```
#define FM_COPY_TGT_BASE_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Copy File Target Filename Error Base ID.

Type: ERROR

Cause

This is the base EID for a number of error events related to the target file in an /FM_COPY command.

Value: 157

Definition at line 1614 of file fm_events.h.

Referenced by FM_CopyFileCmd().

10.5.2.53 FM_COPY_TGT_EXIST_ERR_EID

```
#define FM_COPY_TGT_EXIST_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)
```

FM Child Task Copy File Target File Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a target filename that already exists.

Value: 158

Definition at line 1642 of file fm_events.h.

10.5.2.54 FM_COPY_TGT_INVALID_ERR_EID

```
#define FM_COPY_TGT_INVALID_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Copy File Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with an invalid target filename.

Value: 157

Definition at line 1628 of file fm_events.h.

10.5.2.55 FM_COPY_TGT_ISDIR_ERR_EID

```
#define FM_COPY_TGT_ISDIR_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Copy File Target Filename Is A Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a target filename that is a directory.

Value: 159

Definition at line 1656 of file fm_events.h.

10.5.2.56 FM_COPY_TGT_ISOPEN_ERR_EID

```
#define FM_COPY_TGT_ISOPEN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Copy File Target Filename Exists As Open File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a target filename that is open.

Value: 160

Definition at line 1670 of file fm_events.h.

10.5.2.57 FM_COPY_TGT_UNKNOWN_ERR_EID

```
#define FM_COPY_TGT_UNKNOWN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_UNKNOWN_EID_OFFSET)
```

FM Child Task Copy File Target Filename Unknown State Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a target filename with unknown state.

Value: 162

Definition at line 1684 of file fm_events.h.

10.5.2.58 FM_CREATE_DIR_CHILD_BASE_EID

```
#define FM_CREATE_DIR_CHILD_BASE_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Create Directory Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 259

Definition at line 3054 of file fm_events.h.

Referenced by FM_CreateDirectoryCmd().

10.5.2.59 FM_CREATE_DIR_CHILD_BROKEN_ERR_EID

```
#define FM_CREATE_DIR_CHILD_BROKEN_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_O↩
FFSET)
```

FM Child Task Create Directory Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 261

Definition at line 3110 of file fm_events.h.

10.5.2.60 FM_CREATE_DIR_CHILD_DISABLED_ERR_EID

```
#define FM_CREATE_DIR_CHILD_DISABLED_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_DISABLED_E↩
ID_OFFSET)
```

FM Child Task Create Directory Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 259

Definition at line 3068 of file fm_events.h.

10.5.2.61 FM_CREATE_DIR_CHILD_FULL_ERR_EID

```
#define FM_CREATE_DIR_CHILD_FULL_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFF↩
SET)
```

FM Child Task Create Directory Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 260

Definition at line 3089 of file fm_events.h.

10.5.2.62 FM_CREATE_DIR_CMD_EID

```
#define FM_CREATE_DIR_CMD_EID 55
```

FM Create Directory Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_CreateDir command.

Definition at line 832 of file fm_events.h.

Referenced by FM_ChildCreateDirCmd().

10.5.2.63 FM_CREATE_DIR_OS_ERR_EID

```
#define FM_CREATE_DIR_OS_ERR_EID 57
```

FM Create Directory Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory name is unused and appears to be valid. Refer to the OS specific return value.

Definition at line 858 of file fm_events.h.

Referenced by FM_ChildCreateDirCmd().

10.5.2.64 FM_CREATE_DIR_PKT_ERR_EID

```
#define FM_CREATE_DIR_PKT_ERR_EID 56
```

FM Create Directory Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_CreateDir command packet with an invalid length.

Definition at line 844 of file fm_events.h.

Referenced by FM_CreateDirectoryCmd().

10.5.2.65 FM_CREATE_DIR_SRC_BASE_EID

```
#define FM_CREATE_DIR_SRC_BASE_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Create Directory, Directory Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Create command packet with a directory name that is unusable for one of several reasons.

Value: 253

Definition at line 2998 of file fm_events.h.

Referenced by FM_CreateDirectoryCmd().

10.5.2.66 FM_CREATE_DIR_SRC_DNE_ERR_EID

```
#define FM_CREATE_DIR_SRC_DNE_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Create Directory Name Exists As File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Create command packet with a directory name that exists as a file.

Value: 254

Definition at line 3026 of file fm_events.h.

10.5.2.67 FM_CREATE_DIR_SRC_INVALID_ERR_EID

```
#define FM_CREATE_DIR_SRC_INVALID_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFF↩
SET)
```

FM Child Task Create Directory Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Create command packet with an invalid directory name.

Value: 253

Definition at line 3012 of file fm_events.h.

10.5.2.68 FM_CREATE_DIR_SRC_ISDIR_ERR_EID

```
#define FM_CREATE_DIR_SRC_ISDIR_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Create Directory Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Create command pasket with a directory that already exists.

Value: 255

Definition at line 3040 of file fm_events.h.

10.5.2.69 FM_DECOM_CFE_ERR_EID

```
#define FM_DECOM_CFE_ERR_EID 38
```

FM Decompress File Decompression Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source file exists. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 560 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.70 FM_DECOM_CHILD_BASE_EID

```
#define FM_DECOM_CHILD_BASE_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Decompress File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 226

Definition at line 2588 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.71 FM_DECOM_CHILD_BROKEN_ERR_EID

```
#define FM_DECOM_CHILD_BROKEN_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Decompress File Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 228

Definition at line 2644 of file fm_events.h.

10.5.2.72 FM_DECOM_CHILD_DISABLED_ERR_EID

```
#define FM_DECOM_CHILD_DISABLED_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Decompress File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 226

Definition at line 2602 of file fm_events.h.

10.5.2.73 FM_DECOM_CHILD_FULL_ERR_EID

```
#define FM_DECOM_CHILD_FULL_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Decompress File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 227

Definition at line 2623 of file fm_events.h.

10.5.2.74 FM_DECOM_CMD_EID

```
#define FM_DECOM_CMD_EID 36
```

FM Decompress File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Decompress command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 533 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.75 FM_DECOM_PKT_ERR_EID

```
#define FM_DECOM_PKT_ERR_EID 37
```

FM Decompress File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with an invalid length.

Definition at line 545 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.76 FM_DECOM_SRC_BASE_EID

```
#define FM_DECOM_SRC_BASE_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Decompress File Source Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a source filename that is unusable for one of several reasons.

Value: 214

Definition at line 2461 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.77 FM_DECOM_SRC_DNE_ERR_EID

```
#define FM_DECOM_SRC_DNE_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Decompress File Source Filename Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a source filename that does not exist.

Value: 215

Definition at line 2489 of file fm_events.h.

10.5.2.78 FM_DECOM_SRC_INVALID_ERR_EID

```
#define FM_DECOM_SRC_INVALID_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Decompress File Source Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with an invalid source filename.

Value: 214

Definition at line 2475 of file fm_events.h.

10.5.2.79 FM_DECOM_SRC_ISDIR_ERR_EID

```
#define FM_DECOM_SRC_ISDIR_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Decompress File Source Filename Is A Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a source filename that is a directory.

Value: 216

Definition at line 2503 of file fm_events.h.

10.5.2.80 FM_DECOM_SRC_OPEN_ERR_EID

```
#define FM_DECOM_SRC_OPEN_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Decompress File Source File Is Open Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a source filename that is already open.

Value: 217

Definition at line 2517 of file fm_events.h.

10.5.2.81 FM_DECOM_TGT_BASE_EID

```
#define FM_DECOM_TGT_BASE_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Decompress File Target Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a target filename that is unusable for one of several reasons.

Value: 220

Definition at line 2532 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.82 FM_DECOM_TGT_EXIST_ERR_EID

```
#define FM_DECOM_TGT_EXIST_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)
```

FM Child Task Decompress File Target File Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a target filename that already exists.

Value: 221

Definition at line 2560 of file fm_events.h.

10.5.2.83 FM_DECOM_TGT_INVALID_ERR_EID

```
#define FM_DECOM_TGT_INVALID_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Decompress File Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with an invalid target filename.

Value: 220

Definition at line 2546 of file fm_events.h.

10.5.2.84 FM_DECOM_TGT_ISDIR_ERR_EID

```
#define FM_DECOM_TGT_ISDIR_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Decompress File Target Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Decompress command packet with a target filename that is a directory.

Value: 222

Definition at line 2574 of file fm_events.h.

10.5.2.85 FM_DELETE_ALL_CHILD_BASE_EID

```
#define FM_DELETE_ALL_CHILD_BASE_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Delete All Files Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 211

Definition at line 2390 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.86 FM_DELETE_ALL_CHILD_BROKEN_ERR_EID

```
#define FM_DELETE_ALL_CHILD_BROKEN_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_O←
FFSET)
```

FM Child Task Delete All Files Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 213

Definition at line 2446 of file fm_events.h.

10.5.2.87 FM_DELETE_ALL_CHILD_DISABLED_ERR_EID

```
#define FM_DELETE_ALL_CHILD_DISABLED_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_DISABLED_E↩
ID_OFFSET)
```

FM Child Task Delete All Files Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 211

Definition at line 2404 of file fm_events.h.

10.5.2.88 FM_DELETE_ALL_CHILD_FULL_ERR_EID

```
#define FM_DELETE_ALL_CHILD_FULL_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFF↩
SET)
```

FM Child Task Delete All Files Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 212

Definition at line 2425 of file fm_events.h.

10.5.2.89 FM_DELETE_ALL_CMD_EID

```
#define FM_DELETE_ALL_CMD_EID 31
```

FM Delete All Files Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_DeleteAll command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 465 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.90 FM_DELETE_ALL_FILES_ND_WARNING_EID

```
#define FM_DELETE_ALL_FILES_ND_WARNING_EID 32
```

FM Delete All Files Unable To Delete All Event ID.

Type: INFORMATION

Cause

This general event message is issued if for any reason some files could not be deleted.

Definition at line 477 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.91 FM_DELETE_ALL_OS_ERR_EID

```
#define FM_DELETE_ALL_OS_ERR_EID 35
```

FM Delete All Files Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists. Refer to the OS-specific return value for an indication of what might have caused this error.

Definition at line 516 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.92 FM_DELETE_ALL_PKT_ERR_EID

```
#define FM_DELETE_ALL_PKT_ERR_EID 34
```

FM Delete All Files Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteAll command packet with an invalid length.

Definition at line 501 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.93 FM_DELETE_ALL_SKIP_WARNING_EID

```
#define FM_DELETE_ALL_SKIP_WARNING_EID 33
```

FM Delete All Files Directories Skipped Event ID.

Type: INFORMATION

Cause

This general event message is issued if for any reason a directory skipped when processing a /FM_DeleteAll command.

Definition at line 489 of file fm_events.h.

Referenced by FM_ChildDeleteAllCmd().

10.5.2.94 FM_DELETE_ALL_SRC_BASE_EID

```
#define FM_DELETE_ALL_SRC_BASE_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Delete All Files Directory Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteAll command packet with a directory name that is unusable for one of several reasons.

Value: 205

Definition at line 2334 of file fm_events.h.

Referenced by FM_DeleteAllFilesCmd().

10.5.2.95 FM_DELETE_ALL_SRC_DNE_ERR_EID

```
#define FM_DELETE_ALL_SRC_DNE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Delete All Files Directory Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteAll command packet with a directory name that does not exist.

Value: 206

Definition at line 2362 of file fm_events.h.

10.5.2.96 FM_DELETE_ALL_SRC_FILE_ERR_EID

```
#define FM_DELETE_ALL_SRC_FILE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_ISFILE_EID_OFFSET)
```

FM Child Task Delete All Files Directory Name Is A File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with a directory name that is a file.

Value: 207

Definition at line 2376 of file fm_events.h.

10.5.2.97 FM_DELETE_ALL_SRC_INVALID_ERR_EID

```
#define FM_DELETE_ALL_SRC_INVALID_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFF←
SET)
```

FM Child Task Delete All Files Directory Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with an invalid source filename.

Value: 205

Definition at line 2348 of file fm_events.h.

10.5.2.98 FM_DELETE_CHILD_BASE_EID

```
#define FM_DELETE_CHILD_BASE_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Delete File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 202

Definition at line 2263 of file fm_events.h.

Referenced by FM_DeleteFileCmd().

10.5.2.99 FM_DELETE_CHILD_BROKEN_ERR_EID

```
#define FM_DELETE_CHILD_BROKEN_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Delete File Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 204

Definition at line 2319 of file fm_events.h.

10.5.2.100 FM_DELETE_CHILD_DISABLED_ERR_EID

```
#define FM_DELETE_CHILD_DISABLED_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Delete File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 202

Definition at line 2277 of file fm_events.h.

10.5.2.101 FM_DELETE_CHILD_FULL_ERR_EID

```
#define FM_DELETE_CHILD_FULL_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Delete File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 203

Definition at line 2298 of file fm_events.h.

10.5.2.102 FM_DELETE_CMD_EID

```
#define FM_DELETE_CMD_EID 28
```

FM Delete File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Delete command.

Definition at line 421 of file fm_events.h.

Referenced by FM_ChildDeleteCmd().

10.5.2.103 FM_DELETE_DIR_CHILD_BASE_EID

```
#define FM_DELETE_DIR_CHILD_BASE_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Delete Directory Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 268

Definition at line 3167 of file fm_events.h.

Referenced by FM_DeleteDirectoryCmd().

10.5.2.104 FM_DELETE_DIR_CHILD_BROKEN_ERR_EID

```
#define FM_DELETE_DIR_CHILD_BROKEN_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_O←
FFSET)
```

FM Child Task Delete Directory Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 270

Definition at line 3223 of file fm_events.h.

10.5.2.105 FM_DELETE_DIR_CHILD_DISABLED_ERR_EID

```
#define FM_DELETE_DIR_CHILD_DISABLED_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_DISABLED_E↩
ID_OFFSET)
```

FM Child Task Delete Directory Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 268

Definition at line 3181 of file fm_events.h.

10.5.2.106 FM_DELETE_DIR_CHILD_FULL_ERR_EID

```
#define FM_DELETE_DIR_CHILD_FULL_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFF↩
SET)
```

FM Child Task Delete Directory Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 269

Definition at line 3202 of file fm_events.h.

10.5.2.107 FM_DELETE_DIR_CMD_EID

```
#define FM_DELETE_DIR_CMD_EID 58
```

FM Delete Directory Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_DeleteDir command.

Definition at line 870 of file fm_events.h.

Referenced by FM_ChildDeleteDirCmd().

10.5.2.108 FM_DELETE_DIR_EMPTY_ERR_EID

```
#define FM_DELETE_DIR_EMPTY_ERR_EID 60
```

FM Delete Directory Command Failed Directory Not Empty Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteDir command packet that references a directory that is not empty.

Definition at line 894 of file fm_events.h.

Referenced by FM_ChildDeleteDirCmd().

10.5.2.109 FM_DELETE_DIR_PKT_ERR_EID

```
#define FM_DELETE_DIR_PKT_ERR_EID 59
```

FM Delete Directory Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteDir command packet with an invalid length.

Definition at line 882 of file fm_events.h.

Referenced by FM_DeleteDirectoryCmd().

10.5.2.110 FM_DELETE_DIR_SRC_BASE_EID

```
#define FM_DELETE_DIR_SRC_BASE_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Delete Directory, Directory Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteDir command packet with a directory name that is unusable for one of several reasons.

Value: 262

Definition at line 3125 of file fm_events.h.

Referenced by FM_DeleteDirectoryCmd().

10.5.2.111 FM_DELETE_DIR_SRC_DNE_ERR_EID

```
#define FM_DELETE_DIR_SRC_DNE_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Delete Directory Name Exists As File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteDir command packet with a directory name that exists as a file.

Value: 263

Definition at line 3153 of file fm_events.h.

10.5.2.112 FM_DELETE_DIR_SRC_INVALID_ERR_EID

```
#define FM_DELETE_DIR_SRC_INVALID_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFF↩
SET)
```

FM Child Task Delete Directory Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_DeleteDir command packet with an invalid directory name.

Value: 262

Definition at line 3139 of file fm_events.h.

10.5.2.113 FM_DELETE_OPENDIR_OS_ERR_EID

```
#define FM_DELETE_OPENDIR_OS_ERR_EID 61
```

FM Delete Directory, Direcotry Open Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists and appears to be valid. Refer to the OS specific return values.

Definition at line 908 of file fm_events.h.

Referenced by FM_ChildDeleteDirCmd().

10.5.2.114 FM_DELETE_OS_ERR_EID

```
#define FM_DELETE_OS_ERR_EID 30
```

FM Delete File Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the filename exists and is not open. Refer to the OS-specific return value for an indication of what might have caused this error.

Definition at line 448 of file fm_events.h.

Referenced by FM_ChildDeleteCmd().

10.5.2.115 FM_DELETE_PKT_ERR_EID

```
#define FM_DELETE_PKT_ERR_EID 29
```

FM Delete File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with an invalid length.

Definition at line 433 of file fm_events.h.

Referenced by FM_DeleteFileCmd().

10.5.2.116 FM_DELETE_RMDIR_OS_ERR_EID

```
#define FM_DELETE_RMDIR_OS_ERR_EID 62
```

FM Delete Directory Remove Directory Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists and appears to be valid. Refer to the OS specific return values.

Definition at line 922 of file fm_events.h.

Referenced by FM_ChildDeleteDirCmd().

10.5.2.117 FM_DELETE_SRC_BASE_EID

```
#define FM_DELETE_SRC_BASE_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Delete File Source Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with a source filename that is unusable for one of several reasons.

Value: 196

Definition at line 2193 of file fm_events.h.

Referenced by FM_DeleteFileCmd().

10.5.2.118 FM_DELETE_SRC_DNE_ERR_EID

```
#define FM_DELETE_SRC_DNE_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Delete File Source File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with a source filename that does not exist.

Value: 197

Definition at line 2221 of file fm_events.h.

10.5.2.119 FM_DELETE_SRC_INVALID_ERR_EID

```
#define FM_DELETE_SRC_INVALID_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Delete File Source Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with an invalid source filename.

Value: 196

Definition at line 2207 of file fm_events.h.

10.5.2.120 FM_DELETE_SRC_ISDIR_ERR_EID

```
#define FM_DELETE_SRC_ISDIR_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Delete File Source Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with a source filename that is a directory.

Value: 198

Definition at line 2235 of file fm_events.h.

10.5.2.121 FM_DELETE_SRC_OPEN_ERR_EID

```
#define FM_DELETE_SRC_OPEN_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Delete File File Is Open Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Delete command packet with a source filename that is already open.

Value: 199

Definition at line 2249 of file fm_events.h.

10.5.2.122 FM_EXIT_ERR_EID

```
#define FM_EXIT_ERR_EID 8
```

FM Application Termination Event ID.

Type: ERROR

Cause

This event message is issued when the File Manager application is about to terminate.

Definition at line 150 of file fm_events.h.

Referenced by FM_AppMain().

10.5.2.123 FM_FILE_INFO_CHILD_BASE_EID

```
#define FM_FILE_INFO_CHILD_BASE_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Get File Info Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 250

Definition at line 2927 of file fm_events.h.

Referenced by FM_GetFileInfoCmd().

10.5.2.124 FM_FILE_INFO_CHILD_BROKEN_ERR_EID

```
#define FM_FILE_INFO_CHILD_BROKEN_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFF↩
SET)
```

FM Child Task Get File Info Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 252

Definition at line 2983 of file fm_events.h.

10.5.2.125 FM_FILE_INFO_CHILD_DISABLED_ERR_EID

```
#define FM_FILE_INFO_CHILD_DISABLED_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_DISABLED_EID↩
_OFFSET)
```

FM Child Task Get File Info Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 250

Definition at line 2941 of file fm_events.h.

10.5.2.126 FM_FILE_INFO_CHILD_FULL_ERR_EID

```
#define FM_FILE_INFO_CHILD_FULL_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Get File Info Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 251

Definition at line 2962 of file fm_events.h.

10.5.2.127 FM_FNAME_DNE_EID_OFFSET

```
#define FM_FNAME_DNE_EID_OFFSET 1
```

Definition at line 1520 of file fm_events.h.

Referenced by FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), and FM_VerifyFileExists().

10.5.2.128 FM_FNAME_EXIST_EID_OFFSET

```
#define FM_FNAME_EXIST_EID_OFFSET 1 /* mutually exclusive with DNE */
```

Definition at line 1521 of file fm_events.h.

Referenced by FM_VerifyFileNoExist().

10.5.2.129 FM_FNAME_INVALID_EID_OFFSET

```
#define FM_FNAME_INVALID_EID_OFFSET 0
```

NOTE: From here on, the event IDs will take the form of a "base" EID + an offset. This is done to allow unique event

IDs to be sent from utility functions.

Definition at line 1519 of file fm_events.h.

Referenced by FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_Verify↩FileNoExist(), and FM_VerifyFileNotOpen().

10.5.2.130 FM_FNAME_ISCLOSED_EID_OFFSET

```
#define FM_FNAME_ISCLOSED_EID_OFFSET 4
```

Definition at line 1525 of file fm_events.h.

10.5.2.131 FM_FNAME_ISDIR_EID_OFFSET

```
#define FM_FNAME_ISDIR_EID_OFFSET 2
```

Definition at line 1522 of file fm_events.h.

Referenced by FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), and F↩M_VerifyFileNotOpen().

10.5.2.132 FM_FNAME_ISFILE_EID_OFFSET

```
#define FM_FNAME_ISFILE_EID_OFFSET 2 /* mutually exclusive with ISDIR */
```

Definition at line 1523 of file fm_events.h.

Referenced by FM_VerifyDirExists().

10.5.2.133 FM_FNAME_ISOPEN_EID_OFFSET

```
#define FM_FNAME_ISOPEN_EID_OFFSET 3
```

Definition at line 1524 of file fm_events.h.

Referenced by FM_VerifyFileClosed(), and FM_VerifyFileNotOpen().

10.5.2.134 FM_FNAME_NUM_OFFSETS

```
#define FM_FNAME_NUM_OFFSETS 6
```

Definition at line 1527 of file fm_events.h.

10.5.2.135 FM_FNAME_UNKNOWN_EID_OFFSET

```
#define FM_FNAME_UNKNOWN_EID_OFFSET 5
```

Definition at line 1526 of file fm_events.h.

Referenced by FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_Verify↩FileNoExist(), and FM_VerifyFileNotOpen().

10.5.2.136 FM_GET_DIR_FILE_CHILD_BASE_EID

```
#define FM_GET_DIR_FILE_CHILD_BASE_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Directory List to File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 283

Definition at line 3337 of file fm_events.h.

Referenced by FM_GetDirListFileCmd().

10.5.2.137 FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID

```
#define FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_BROKEN_E↩
ID_OFFSET)
```

FM Child Task Directory List to File Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 285

Definition at line 3393 of file fm_events.h.

10.5.2.138 FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID

```
#define FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_DISABL↩
ED_EID_OFFSET)
```

FM Child Task Directory List to File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 283

Definition at line 3351 of file fm_events.h.

10.5.2.139 FM_GET_DIR_FILE_CHILD_FULL_ERR_EID

```
#define FM_GET_DIR_FILE_CHILD_FULL_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID↩
_OFFSET)
```

FM Child Task Directory List to File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 284

Definition at line 3372 of file fm_events.h.

10.5.2.140 FM_GET_DIR_FILE_CMD_EID

```
#define FM_GET_DIR_FILE_CMD_EID 63
```

FM Directory List To File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_GetDirFile command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 939 of file fm_events.h.

Referenced by FM_ChildDirListFileLoop().

10.5.2.141 FM_GET_DIR_FILE_OSCREAT_ERR_EID

```
#define FM_GET_DIR_FILE_OSCREAT_ERR_EID 69
```

FM Directory List To File Create File Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that prevents the output file from being created. This error occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 1040 of file fm_events.h.

Referenced by FM_ChildDirListFileInit().

10.5.2.142 FM_GET_DIR_FILE_OSOPENDIR_ERR_EID

```
#define FM_GET_DIR_FILE_OSOPENDIR_ERR_EID 66
```

FM Directory List To File Directory Open Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 986 of file fm_events.h.

Referenced by FM_ChildDirListFileCmd().

10.5.2.143 FM_GET_DIR_FILE_PKT_ERR_EID

```
#define FM_GET_DIR_FILE_PKT_ERR_EID 64
```

FM Directory List To File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with an invalid length.

Definition at line 951 of file fm_events.h.

Referenced by FM_GetDirListFileCmd().

10.5.2.144 FM_GET_DIR_FILE_SRC_BASE_EID

```
#define FM_GET_DIR_FILE_SRC_BASE_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Directory List to File Source Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with a source directory name that is unusable for one of several reasons.

Value: 271

Definition at line 3238 of file fm_events.h.

Referenced by FM_GetDirListFileCmd().

10.5.2.145 FM_GET_DIR_FILE_SRC_DNE_ERR_EID

```
#define FM_GET_DIR_FILE_SRC_DNE_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Directory List to File Directory Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with a source directory name that does not exist.

Value: 272

Definition at line 3266 of file fm_events.h.

10.5.2.146 FM_GET_DIR_FILE_SRC_INVALID_ERR_EID

```
#define FM_GET_DIR_FILE_SRC_INVALID_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_INVALID_EID↩
_OFFSET)
```

FM Child Task Directory List to File Directory Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with an invalid source directory name.

Value: 271

Definition at line 3252 of file fm_events.h.

10.5.2.147 FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID

```
#define FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_ISFILE_EID_OF↩
FSET)
```

FM Child Task Directory List to File Directory Name Is File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with a source directory name that is a file.

Value: 273

Definition at line 3280 of file fm_events.h.

10.5.2.148 FM_GET_DIR_FILE_TGT_BASE_EID

```
#define FM_GET_DIR_FILE_TGT_BASE_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Directory List to File Target Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with a target filename that is unusable for one of several reasons.

Value: 277

Definition at line 3295 of file fm_events.h.

Referenced by FM_GetDirListFileCmd().

10.5.2.149 FM_GET_DIR_FILE_TGT_INVALID_ERR_EID

```
#define FM_GET_DIR_FILE_TGT_INVALID_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_INVALID_EID↩
_OFFSET)
```

FM Child Task Directory List to File Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with an invalid target file name.

Value: 277

Definition at line 3309 of file fm_events.h.

10.5.2.150 FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID

```
#define FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFF↩
SET)
```

FM Child Task Directory List to File Target Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirFile command packet with a target filename that is a directory.

Value: 279

Definition at line 3323 of file fm_events.h.

10.5.2.151 FM_GET_DIR_FILE_UPSTATS_ERR_EID

```
#define FM_GET_DIR_FILE_UPSTATS_ERR_EID 71
```

FM Directory List To File Write Update Stats Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that prevents updated statistics from being written to a file. This occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 1076 of file fm_events.h.

Referenced by FM_ChildDirListFileLoop().

10.5.2.152 FM_GET_DIR_FILE_WARNING_EID

```
#define FM_GET_DIR_FILE_WARNING_EID 65
```

FM Directory List To File Command Combined Path and Name Too Long Event ID.

Type: INFORMATION

Cause

This event message is generated when the combined length of the directory name plus the directory entry name exceeds the maximum qualified filename length. It is unclear how this condition might arise, but since we are copying both strings into a fixed length buffer, we must first verify the length.

The /FM_GetDirFile command handler will not write information regarding this directory entry to the output file.

Definition at line 969 of file fm_events.h.

Referenced by FM_ChildDirListFileLoop().

10.5.2.153 FM_GET_DIR_FILE_WRBLANK_ERR_EID

```
#define FM_GET_DIR_FILE_WRBLANK_ERR_EID 67
```

FM Directory List To File Write Blank Stats Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an error when writing a blank stats structure using the OS_write function. This occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 1004 of file fm_events.h.

Referenced by FM_ChildDirListFileInit().

10.5.2.154 FM_GET_DIR_FILE_WRENTRY_ERR_EID

```
#define FM_GET_DIR_FILE_WRENTRY_ERR_EID 70
```

FM Directory List To File Write Entry Failed Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that prevents an entry from being written. This error occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 1058 of file fm_events.h.

Referenced by FM_ChildDirListFileLoop().

10.5.2.155 FM_GET_DIR_FILE_WRHDR_ERR_EID

```
#define FM_GET_DIR_FILE_WRHDR_ERR_EID 68
```

FM Directory List To File Write Header Failed Event ID.

Type: ERROR

Cause

This event message is generated when the header cannot be written to the file using CFE_FS_WriteHeader. This error occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values.

Definition at line 1022 of file fm_events.h.

Referenced by FM_ChildDirListFileInit().

10.5.2.156 FM_GET_DIR_PKT_CHILD_BASE_EID

```
#define FM_GET_DIR_PKT_CHILD_BASE_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Directory List to Packet Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 292

Definition at line 3464 of file fm_events.h.

Referenced by FM_GetDirListPktCmd().

10.5.2.157 FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID

```
#define FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CHILD_BROKEN_EID↩
_OFFSET)
```

FM Child Task Directory List to Packet Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 294

Definition at line 3520 of file fm_events.h.

10.5.2.158 FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID

```
#define FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CHILD_DISABLED↩
_EID_OFFSET)
```

FM Child Task Directory List to Packet Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 292

Definition at line 3478 of file fm_events.h.

10.5.2.159 FM_GET_DIR_PKT_CHILD_FULL_ERR_EID

```
#define FM_GET_DIR_PKT_CHILD_FULL_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_O↩
FFSET)
```

FM Child Task Directory List to Packet Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 293

Definition at line 3499 of file fm_events.h.

10.5.2.160 FM_GET_DIR_PKT_CMD_EID

```
#define FM_GET_DIR_PKT_CMD_EID 72
```

FM Directory List To Packet Command Event ID.

Type: DEBUG

This event is type debug because the command generates a telemetry packet that also signals the completion of the command.

Cause

This event message signals the successful completion of a /FM_GetDirPkt command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 1096 of file fm_events.h.

Referenced by FM_ChildDirListPktCmd().

10.5.2.161 FM_GET_DIR_PKT_OS_ERR_EID

```
#define FM_GET_DIR_PKT_OS_ERR_EID 75
```

FM Directory List To Packet Directory Open Failed Event ID.

Type: ERROR

Cause

The numeric data in the event is the return value from the OS function call. The string data identifies the name of the directory or the directory entry.

Definition at line 1136 of file fm_events.h.

Referenced by FM_ChildDirListPktCmd().

10.5.2.162 FM_GET_DIR_PKT_PKT_ERR_EID

```
#define FM_GET_DIR_PKT_PKT_ERR_EID 74
```

FM Directory List To Packet Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirPkt command packet with an invalid length.

Definition at line 1123 of file fm_events.h.

Referenced by FM_GetDirListPktCmd().

10.5.2.163 FM_GET_DIR_PKT_SRC_BASE_EID

```
#define FM_GET_DIR_PKT_SRC_BASE_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Directory List to Packet Directory Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirPkt command packet with a source directory name that is unusable for one of several reasons.

Value: 286

Definition at line 3408 of file fm_events.h.

Referenced by FM_GetDirListPktCmd().

10.5.2.164 FM_GET_DIR_PKT_SRC_DNE_ERR_EID

```
#define FM_GET_DIR_PKT_SRC_DNE_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Directory List to Packet Directory Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirPkt command packet with a source directory name that does not exist.

Value: 287

Definition at line 3436 of file fm_events.h.

10.5.2.165 FM_GET_DIR_PKT_SRC_INVALID_ERR_EID

```
#define FM_GET_DIR_PKT_SRC_INVALID_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_INVALID_EID_O←
FFSET)
```

FM Child Task Directory List to Packet Directory Name Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirPkt command packet with an invalid source directory name.

Value: 286

Definition at line 3422 of file fm_events.h.

10.5.2.166 FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID

```
#define FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_ISFILE_EID_OFFS←
ET)
```

FM Child Task Directory List to Packet Directory Is File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetDirPkt command packet with a source directory name that is a file.

Value: 288

Definition at line 3450 of file fm_events.h.

10.5.2.167 FM_GET_DIR_PKT_WARNING_EID

```
#define FM_GET_DIR_PKT_WARNING_EID 73
```

FM Directory List To Packet Command Directory and Entry Too Long Event ID.

Type: INFORMATION

Cause

This event message is generated when the combined length of the directory name plus the directory entry name exceeds the maximum qualified filename length. It is unclear how this condition might arise, but since we are copying both strings into a fixed length buffer, we must first verify the length.

Definition at line 1111 of file fm_events.h.

Referenced by FM_ChildDirListPktCmd().

10.5.2.168 FM_GET_FILE_INFO_CMD_EID

```
#define FM_GET_FILE_INFO_CMD_EID 46
```

FM Get File Info Command Event ID.

Type: DEBUG

This event is type debug because the command generates a telemetry packet that also signals the completion of the command.

Cause

This event message signals the successful completion of a /FM_GetFileInfo command.

Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus this event may not occur until some time after the command was invoked. However, this event message does signal the actual completion of the command.

Definition at line 697 of file fm_events.h.

Referenced by FM_ChildFileInfoCmd().

10.5.2.169 FM_GET_FILE_INFO_OPEN_ERR_EID

```
#define FM_GET_FILE_INFO_OPEN_ERR_EID 49
```

FM Get File Info Unable To Compute CRC File Open Failed Event ID.

Type: ERROR

Cause

This event message is generated when the CRC of a file cannot be computed because the file cannot be opened.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 751 of file fm_events.h.

Referenced by FM_ChildFileInfoCmd().

10.5.2.170 FM_GET_FILE_INFO_PKT_ERR_EID

```
#define FM_GET_FILE_INFO_PKT_ERR_EID 51
```

FM Get File Info Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetFileInfo command packet with an invalid length.

Definition at line 781 of file fm_events.h.

Referenced by FM_GetFileInfoCmd().

10.5.2.171 FM_GET_FILE_INFO_READ_WARNING_EID

```
#define FM_GET_FILE_INFO_READ_WARNING_EID 50
```

FM Get File Info Unable To Compute CRC File Read Failed Event ID.

Type: INFORMATION

Cause

This event message is generated when the CRC of a file cannot be computed because the file cannot be read.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 769 of file fm_events.h.

Referenced by FM_ChildFileInfoCmd().

10.5.2.172 FM_GET_FILE_INFO_SRC_ERR_EID

```
#define FM_GET_FILE_INFO_SRC_ERR_EID 52
```

FM Get File Info Command Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetFileInfo command packet with an invalid filename.

Definition at line 793 of file fm_events.h.

Referenced by FM_GetFileInfoCmd().

10.5.2.173 FM_GET_FILE_INFO_STATE_WARNING_EID

```
#define FM_GET_FILE_INFO_STATE_WARNING_EID 47
```

FM Get File Info Unable To Compute CRC File State Invalid Event ID.

Type: INFORMATION

Cause

This event message is generated when the CRC of a file cannot be computed because the file has an invalid state.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 715 of file fm_events.h.

Referenced by FM_ChildFileInfoCmd().

10.5.2.174 FM_GET_FILE_INFO_TYPE_WARNING_EID

```
#define FM_GET_FILE_INFO_TYPE_WARNING_EID 48
```

FM Get File Info Unable To Compute CRC, CRC Type Invalid Event ID.

Type: INFORMATION

Cause

This event message is generated when the CRC of a file cannot be computed because the CRC type is invalid.

This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Definition at line 733 of file fm_events.h.

Referenced by FM_ChildFileInfoCmd().

10.5.2.175 FM_GET_FREE_SPACE_CMD_EID

```
#define FM_GET_FREE_SPACE_CMD_EID 76
```

FM Get Free Space Command Event ID.

Type: DEBUG

This event is type debug because the command generates a telemetry packet that also signals the completion of the command.

Cause

This event message signals the successful completion of a /FM_GetFreeSpace command.

Definition at line 1151 of file fm_events.h.

Referenced by FM_GetFreeSpaceCmd().

10.5.2.176 FM_GET_FREE_SPACE_PKT_ERR_EID

```
#define FM_GET_FREE_SPACE_PKT_ERR_EID 77
```

FM Get Free Space Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetFreeSpace command packet with an invalid length.

Definition at line 1163 of file fm_events.h.

Referenced by FM_GetFreeSpaceCmd().

10.5.2.177 FM_GET_FREE_SPACE_TBL_ERR_EID

```
#define FM_GET_FREE_SPACE_TBL_ERR_EID 78
```

FM Get Free Space Table Not Loaded Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetFreeSpace command packet when the FM file system free space table has not yet been loaded.

Definition at line 1176 of file fm_events.h.

Referenced by FM_GetFreeSpaceCmd().

10.5.2.178 FM_GET_OPEN_FILES_CMD_EID

```
#define FM_GET_OPEN_FILES_CMD_EID 53
```

FM Get Open Files Command Event ID.

Type: DEBUG

This event is type debug because the command generates a telemetry packet that also signals the completion of the command.

Cause

This event message signals the successful completion of a /FM_GetOpenFiles command.

Definition at line 808 of file fm_events.h.

Referenced by FM_GetOpenFilesCmd().

10.5.2.179 FM_GET_OPEN_FILES_PKT_ERR_EID

```
#define FM_GET_OPEN_FILES_PKT_ERR_EID 54
```

FM Get Open Files Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_GetOpenFiles command packet with an invalid length.

Definition at line 820 of file fm_events.h.

Referenced by FM_GetOpenFilesCmd().

10.5.2.180 FM_HK_REQ_ERR_EID

```
#define FM_HK_REQ_ERR_EID 11
```

FM Command Packet Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a housekeeping request command packet with an invalid length.

Definition at line 191 of file fm_events.h.

Referenced by FM_ReportHK().

10.5.2.181 FM_MID_ERR_EID

```
#define FM_MID_ERR_EID 9
```

FM Main Loop Message ID Invalid Event ID.

Type: ERROR

Cause

This event message is issued when the File Manager application has received an unexpected Software Bus packet. There is no obvious explanation of why or how FM could receive such a packet.

Definition at line 163 of file fm_events.h.

Referenced by FM_ProcessPkt().

10.5.2.182 FM_MOVE_CHILD_BASE_EID

```
#define FM_MOVE_CHILD_BASE_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Move File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 178

Definition at line 1910 of file fm_events.h.

Referenced by FM_MoveFileCmd().

10.5.2.183 FM_MOVE_CHILD_BROKEN_ERR_EID

```
#define FM_MOVE_CHILD_BROKEN_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Move File Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 180

Definition at line 1966 of file fm_events.h.

10.5.2.184 FM_MOVE_CHILD_DISABLED_ERR_EID

```
#define FM_MOVE_CHILD_DISABLED_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Move File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 178

Definition at line 1924 of file fm_events.h.

10.5.2.185 FM_MOVE_CHILD_FULL_ERR_EID

```
#define FM_MOVE_CHILD_FULL_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Move File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 179

Definition at line 1945 of file fm_events.h.

10.5.2.186 FM_MOVE_CMD_EID

```
#define FM_MOVE_CMD_EID 20
```

FM Move File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Move command.

Definition at line 313 of file fm_events.h.

Referenced by FM_ChildMoveCmd().

10.5.2.187 FM_MOVE_OS_ERR_EID

```
#define FM_MOVE_OS_ERR_EID 23
```

FM Move File Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. Verify that the target filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return value.

Definition at line 355 of file fm_events.h.

Referenced by FM_ChildMoveCmd().

10.5.2.188 FM_MOVE_OVR_ERR_EID

```
#define FM_MOVE_OVR_ERR_EID 22
```

FM Move File Command Overwrite Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with an invalid overwrite argument. Overwrite must be set to TRUE (one) or FALSE (zero).

Definition at line 338 of file fm_events.h.

Referenced by FM_MoveFileCmd().

10.5.2.189 FM_MOVE_PKT_ERR_EID

```
#define FM_MOVE_PKT_ERR_EID 21
```

FM Move File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with an invalid length.

Definition at line 325 of file fm_events.h.

Referenced by FM_MoveFileCmd().

10.5.2.190 FM_MOVE_SRC_BASE_EID

```
#define FM_MOVE_SRC_BASE_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Move File Source Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a source filename that is unusable for one of several reasons.

Value: 166

Definition at line 1769 of file fm_events.h.

Referenced by FM_MoveFileCmd().

10.5.2.191 FM_MOVE_SRC_DNE_ERR_EID

```
#define FM_MOVE_SRC_DNE_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Move File Source File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a source filename that does not exist.

Value: 167

Definition at line 1797 of file fm_events.h.

10.5.2.192 FM_MOVE_SRC_INVALID_ERR_EID

```
#define FM_MOVE_SRC_INVALID_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Move File Source Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with an invalid source filename.

Value: 166

Definition at line 1783 of file fm_events.h.

10.5.2.193 FM_MOVE_SRC_ISDIR_ERR_EID

```
#define FM_MOVE_SRC_ISDIR_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Move File Source Filename Is A Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a source filename that is a directory.

Value: 168

Definition at line 1811 of file fm_events.h.

10.5.2.194 FM_MOVE_TGT_BASE_EID

```
#define FM_MOVE_TGT_BASE_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Move File Target Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a target filename that is unusable for one of several reasons.

Value: 172

Definition at line 1826 of file fm_events.h.

Referenced by FM_MoveFileCmd().

10.5.2.195 FM_MOVE_TGT_EXIST_ERR_EID

```
#define FM_MOVE_TGT_EXIST_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)
```

FM Child Task Move File Target File Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a target filename that already exists

Value: 173

Definition at line 1854 of file fm_events.h.

10.5.2.196 FM_MOVE_TGT_INVALID_ERR_EID

```
#define FM_MOVE_TGT_INVALID_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Move File Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with an invalid target filename.

Value: 172

Definition at line 1840 of file fm_events.h.

10.5.2.197 FM_MOVE_TGT_ISDIR_ERR_EID

```
#define FM_MOVE_TGT_ISDIR_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Move File Target Filename Is A Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a target filename that is a directory.

Value: 174

Definition at line 1868 of file fm_events.h.

10.5.2.198 FM_MOVE_TGT_ISOPEN_ERR_EID

```
#define FM_MOVE_TGT_ISOPEN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Move File Target File Exists As An Open File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Move command packet with a target filename that is open.

Value: 175

Definition at line 1882 of file fm_events.h.

10.5.2.199 FM_MOVE_TGT_UNKNOWN_ERR_EID

```
#define FM_MOVE_TGT_UNKNOWN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_UNKNOWN_EID_OFFSET)
```

FM Child Task Move File Target File Has Unknown State Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Copy command packet with a target file with unknown state..

Value: 177

Definition at line 1896 of file fm_events.h.

10.5.2.200 FM_NOOP_CMD_EID

```
#define FM_NOOP_CMD_EID 12
```

FM No-op Command Event ID.

Type: INFORMATION

Cause

This event message signals the successful completion of a /FM_Noop command.

Definition at line 203 of file fm_events.h.

Referenced by FM_NoopCmd().

10.5.2.201 FM_NOOP_PKT_ERR_EID

```
#define FM_NOOP_PKT_ERR_EID 13
```

FM No-op Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Noop command packet with an invalid length.

Definition at line 215 of file fm_events.h.

Referenced by FM_NoopCmd().

10.5.2.202 FM_OS_SYS_STAT_ERR_EID

```
#define FM_OS_SYS_STAT_ERR_EID 103
```

FM Get Free Space Get File SYstem Stats Failed Event ID.

Type: ERROR

Cause:

This event message occurs if the free space for a file system cannot be read when processing the FM_GetFreeSpace↩Cmd command.

Definition at line 1511 of file fm_events.h.

Referenced by FM_GetFreeSpaceCmd().

10.5.2.203 FM_RENAME_CHILD_BASE_EID

```
#define FM_RENAME_CHILD_BASE_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Rename File Child Task Error Base ID.

Type: ERROR

Cause

This is the base for any of several messages that are generated when the FM child task command queue interface cannot be used.

Value: 193

Definition at line 2122 of file fm_events.h.

Referenced by FM_RenameFileCmd().

10.5.2.204 FM_RENAME_CHILD_BROKEN_ERR_EID

```
#define FM_RENAME_CHILD_BROKEN_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)
```

FM Child Task Rename File Child Task Inteface Broken Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the interface between the main task and child task is broken.

If the child task queue is broken then either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Value: 195

Definition at line 2178 of file fm_events.h.

10.5.2.205 FM_RENAME_CHILD_DISABLED_ERR_EID

```
#define FM_RENAME_CHILD_DISABLED_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)
```

FM Child Task Rename File Child Task Disabled Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task is disabled.

Value: 193

Definition at line 2136 of file fm_events.h.

10.5.2.206 FM_RENAME_CHILD_FULL_ERR_EID

```
#define FM_RENAME_CHILD_FULL_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)
```

FM Child Task Rename File Child Task Queue Full Event ID.

Type: ERROR

Cause

This event message is generated when the FM child task command queue interface cannot be used because the child task comand queue is full.

If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use CFE commands to terminate the child task, which should then cause FM to process all commands in the main task.

Value: 194

Definition at line 2157 of file fm_events.h.

10.5.2.207 FM_RENAME_CMD_EID

```
#define FM_RENAME_CMD_EID 24
```

FM Rename File Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_Rename command.

Definition at line 367 of file fm_events.h.

Referenced by FM_ChildRenameCmd().

10.5.2.208 FM_RENAME_OS_ERR_EID

```
#define FM_RENAME_OS_ERR_EID 27
```

FM Rename File Command OS Error Event ID.

Type: ERROR

Cause

This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. Verify that the target filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return value.

Definition at line 409 of file fm_events.h.

Referenced by FM_ChildRenameCmd().

10.5.2.209 FM_RENAME_OVR_ERR_EID

```
#define FM_RENAME_OVR_ERR_EID 26
```

FM Rename File Command Overwrite Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with an invalid overwrite argument. Overwrite must be set to TRUE (one) or FALSE (zero).

Definition at line 392 of file fm_events.h.

10.5.2.210 FM_RENAME_PKT_ERR_EID

```
#define FM_RENAME_PKT_ERR_EID 25
```

FM Rename File Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with an invalid length.

Definition at line 379 of file fm_events.h.

Referenced by FM_RenameFileCmd().

10.5.2.211 FM_RENAME_SRC_BASE_EID

```
#define FM_RENAME_SRC_BASE_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)
```

FM Child Task Rename File Source Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a source filename that is unusable for one of several reasons.

Value: 181

Definition at line 1981 of file fm_events.h.

Referenced by FM_RenameFileCmd().

10.5.2.212 FM_RENAME_SRC_DNE_ERR_EID

```
#define FM_RENAME_SRC_DNE_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)
```

FM Child Task Rename File Source File Does Not Exist Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a source filename that does not exist.

Value: 182

Definition at line 2009 of file fm_events.h.

10.5.2.213 FM_RENAME_SRC_INVALID_ERR_EID

```
#define FM_RENAME_SRC_INVALID_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Rename File Source Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with an invalid source filename.

Value: 181

Definition at line 1995 of file fm_events.h.

10.5.2.214 FM_RENAME_SRC_ISDIR_ERR_EID

```
#define FM_RENAME_SRC_ISDIR_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Rename File Source Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a source filename that is a directory.

Value: 183

Definition at line 2023 of file fm_events.h.

10.5.2.215 FM_RENAME_TGT_BASE_EID

```
#define FM_RENAME_TGT_BASE_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)
```

FM Child Task Rename File Target Filename Error Base ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a target filename that is unusable for one of several reasons.

Value: 187

Definition at line 2038 of file fm_events.h.

Referenced by FM_RenameFileCmd().

10.5.2.216 FM_RENAME_TGT_EXIST_ERR_EID

```
#define FM_RENAME_TGT_EXIST_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)
```

FM Child Task Rename File Target File Already Exists Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a target filename that already exists.

Value: 188

Definition at line 2066 of file fm_events.h.

10.5.2.217 FM_RENAME_TGT_INVALID_ERR_EID

```
#define FM_RENAME_TGT_INVALID_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)
```

FM Child Task Rename File Target Filename Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with an invalid target filename.

Value: 187

Definition at line 2052 of file fm_events.h.

10.5.2.218 FM_RENAME_TGT_ISDIR_ERR_EID

```
#define FM_RENAME_TGT_ISDIR_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)
```

FM Child Task Rename File Target Filename Is Directory Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a target filename that is a directory.

Value: 189

Definition at line 2080 of file fm_events.h.

10.5.2.219 FM_RENAME_TGT_ISOPEN_ERR_EID

```
#define FM_RENAME_TGT_ISOPEN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)
```

FM Child Task Rename File Target Filename Exists As Open File Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a target filename that is open.

Value: 190

Definition at line 2094 of file fm_events.h.

10.5.2.220 FM_RENAME_TGT_UNKNOWN_ERR_EID

```
#define FM_RENAME_TGT_UNKNOWN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_UNKNOWN_EID_OFFSET)
```

FM Child Task Rename File Target Filename Has Unknown State Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_Rename command packet with a target file with unknown state.

Value: 192

Definition at line 2108 of file fm_events.h.

10.5.2.221 FM_RESET_CMD_EID

```
#define FM_RESET_CMD_EID 14
```

FM Reset Counters Command Event ID.

Type: DEBUG

This event is type debug because the command resets housekeeping telemetry counters that also signal the completion of the command.

Cause

This event message signals the successful completion of a /FM_ResetCtrs command.

Definition at line 230 of file fm_events.h.

Referenced by FM_ResetCountersCmd().

10.5.2.222 FM_RESET_PKT_ERR_EID

```
#define FM_RESET_PKT_ERR_EID 15
```

FM Reset Counters Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_ResetCtrs command packet with an invalid length.

Definition at line 242 of file fm_events.h.

Referenced by FM_ResetCountersCmd().

10.5.2.223 FM_SB_RECEIVE_ERR_EID

```
#define FM_SB_RECEIVE_ERR_EID 7
```

FM Main Loop Receive from Software Bus Failed Event ID.

Type: ERROR

Cause

This event message is issued when the File Manager application has failed in its attempt to read from its Software Bus input pipe while processing the software main loop sequence.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 138 of file fm_events.h.

Referenced by FM_AppMain().

10.5.2.224 FM_SB_RECEIVE_NULL_PTR_ERR_EID

```
#define FM_SB_RECEIVE_NULL_PTR_ERR_EID 102
```

FM Main Loop Software Bus Returned NULL On Success Event ID.

Type: ERROR

Cause

This event message occurs if the Software Bus returns a success status in the main loop but provided a NULL pointer as the return argument.

Definition at line 1499 of file fm_events.h.

Referenced by FM_AppMain().

10.5.2.225 FM_SET_PERM_CMD_EID

```
#define FM_SET_PERM_CMD_EID 99
```

FM Set Permissions Command Event ID.

Type: DEBUG

Cause

This event message signals the successful completion of a /FM_SetPerm command.

Definition at line 1463 of file fm_events.h.

Referenced by FM_ChildSetPermissionsCmd().

10.5.2.226 FM_SET_PERM_ERR_EID

```
#define FM_SET_PERM_ERR_EID 98
```

FM Set Permissions Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_SetPermissions command packet with an invalid length.

Definition at line 1451 of file fm_events.h.

Referenced by FM_SetPermissionsCmd().

10.5.2.227 FM_SET_PERM_OS_ERR_EID

```
#define FM_SET_PERM_OS_ERR_EID 100
```

FM Set Permissions Command Chmod Error Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_SetPerm command packet with an invalid length.

Definition at line 1475 of file fm_events.h.

Referenced by FM_ChildSetPermissionsCmd().

10.5.2.228 FM_SET_TABLE_STATE_ARG_IDX_ERR_EID

```
#define FM_SET_TABLE_STATE_ARG_IDX_ERR_EID 82
```

FM Set Table State Command Index Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a FM_SetTableState command packet with an invalid table index argument.

Definition at line 1225 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.229 FM_SET_TABLE_STATE_ARG_STATE_ERR_EID

```
#define FM_SET_TABLE_STATE_ARG_STATE_ERR_EID 83
```

FM Set Table State Command State Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a FM_SetTableState command packet with an invalid entry state argument.

Definition at line 1237 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.230 FM_SET_TABLE_STATE_CMD_EID

```
#define FM_SET_TABLE_STATE_CMD_EID 79
```

FM Set Table State Command Event ID.

Type: INFORMATION

Cause

This event message signals the successful completion of a /FM_SetTableState command.

Definition at line 1188 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.231 FM_SET_TABLE_STATE_PKT_ERR_EID

```
#define FM_SET_TABLE_STATE_PKT_ERR_EID 80
```

FM Set Table State Command Length Invalid Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_SetTableState command packet with an invalid length.

Definition at line 1200 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.232 FM_SET_TABLE_STATE_TBL_ERR_EID

```
#define FM_SET_TABLE_STATE_TBL_ERR_EID 81
```

FM Set Table State Command Table Not Loaded Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_SetTableState command packet when the FM file system free space table has not yet been loaded.

Definition at line 1213 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.233 FM_SET_TABLE_STATE_UNUSED_ERR_EID

```
#define FM_SET_TABLE_STATE_UNUSED_ERR_EID 84
```

FM Set Table State Command Unused Entry Event ID.

Type: ERROR

Cause

This event message is generated upon receipt of a /FM_SetTableState command packet that references an unused free space table entry.

Definition at line 1249 of file fm_events.h.

Referenced by FM_SetTableStateCmd().

10.5.2.234 FM_STARTUP_CREAT_PIPE_ERR_EID

```
#define FM_STARTUP_CREAT_PIPE_ERR_EID 3
```

FM Initialization Create SB Input Pipe Failed Event ID.

Type: Error

Cause

This event message is issued when the File Manager application has failed in its attempt to create a Software Bus input pipe during startup initialization.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 74 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.235 FM_STARTUP_EID

```
#define FM_STARTUP_EID 1
```

FM Initialization Event ID.

Type: INFORMATION

Cause

This event message is issued after the File Manager application has successfully completed startup initialization.

Definition at line 42 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.236 FM_STARTUP_EVENTS_ERR_EID

```
#define FM_STARTUP_EVENTS_ERR_EID 2
```

FM Initialization Register For Event Services Failed Event ID.

Type: Error

Cause

This event message is issued when the File Manager application has failed in its attempt to register for event services during startup initialization.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 58 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.237 FM_STARTUP_SUBSCRIB_GCMD_ERR_EID

```
#define FM_STARTUP_SUBSCRIB_GCMD_ERR_EID 5
```

FM Initialization Subscribe to FM Commands Failed Event ID.

Type: Error

Cause

This event message is issued when the File Manager application has failed in its attempt to subscribe to the FM ground command packet during startup initialization.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 106 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.238 FM_STARTUP_SUBSCRIB_HK_ERR_EID

```
#define FM_STARTUP_SUBSCRIB_HK_ERR_EID 4
```

FM Initialization Subscribe to HK Request Failed Event ID.

Type: Error

Cause

This event message is issued when the File Manager application has failed in its attempt to subscribe to the HK telemetry request command during startup initialization.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 90 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.239 FM_STARTUP_TABLE_INIT_ERR_EID

```
#define FM_STARTUP_TABLE_INIT_ERR_EID 6
```

FM Initialization Register Free Space Table Failed Event ID.

Type: Error

Cause

This event message is issued when the File Manager application has failed in its attempt to register its file system free space table during startup initialization.

This is a fatal error that will cause the File Manager application to terminate.

Definition at line 122 of file fm_events.h.

Referenced by FM_AppInit().

10.5.2.240 FM_TABLE_VERIFY_BAD_STATE_ERR_EID

```
#define FM_TABLE_VERIFY_BAD_STATE_ERR_EID 88
```

FM Free Space Table Verification Failed State Invalid Event ID.

Type: ERROR

Cause

This event message is generated when a file system free space table fails the table verification process because a table entry has an invalid state. Each file system table entry has only 2 fields: table entry state and file system name. The table entry state field must be either enabled or disabled. The file system name string must have a non-zero length, include a string terminator and not contain characters considered invalid for filenames.

If the file system free space table loaded at startup fails verification, the FM application will not terminate. However, the FM application will not process commands that request the file system free space telemetry packet if a file system free space table has not been successfully loaded. Thereafter, if an attempt to load a new table fails verification, the FM application will continue to use the previous table.

Definition at line 1318 of file fm_events.h.

Referenced by FM_ValidateTable().

10.5.2.241 FM_TABLE_VERIFY_EID

```
#define FM_TABLE_VERIFY_EID 97
```

FM Free Space Table Validation Results Event ID.

Type: INFORMATION

Cause:

This event describes the results of the Free Space Table validation function. The cFE Table Services Manager will call this function autonomously when the default table is loaded at startup and also whenever a table validate command (that targets this table) is processed.

Definition at line 1439 of file fm_events.h.

Referenced by FM_ValidateTable().

10.5.2.242 FM_TABLE_VERIFY_EMPTY_ERR_EID

```
#define FM_TABLE_VERIFY_EMPTY_ERR_EID 85
```

FM Free Space Table Verification Failed Empty Name Event ID.

Type: ERROR

Cause

This event message is generated when a file system free space table fails the table verification process because the file system name is an empty string. Each file system table entry has only 2 fields: table entry state and file system name. The table entry state field must be either enabled or disabled. The file system name string must have a non-zero length, include a string terminator and not contain characters considered invalid for filenames.

If the file system free space table loaded at startup fails verification, the FM application will not terminate. However, the FM application will not process commands that request the file system free space telemetry packet if a file system free space table has not been successfully loaded. Thereafter, if an attempt to load a new table fails verification, the FM application will continue to use the previous table.

Definition at line 1272 of file fm_events.h.

Referenced by FM_ValidateTable().

10.5.2.243 FM_TABLE_VERIFY_NULL_PTR_ERR_EID

```
#define FM_TABLE_VERIFY_NULL_PTR_ERR_EID 101
```

FM Free Space Table Verification Failed Null Pointer Detected.

Type: ERROR

Cause

This event message occurs when the FM validate table callback function receives a NULL pointer as the input argument.

Definition at line 1487 of file fm_events.h.

Referenced by FM_ValidateTable().

10.5.2.244 FM_TABLE_VERIFY_TOOLONG_ERR_EID

```
#define FM_TABLE_VERIFY_TOOLONG_ERR_EID 86
```

FM Free Space Table Verification Failed Name Too Long Event ID.

Type: ERROR

Cause

This event message is generated when a file system free space table fails the table verification process because the file system name is too long. Each file system table entry has only 2 fields: table entry state and file system name. The table entry state field must be either enabled or disabled. The file system name string must have a non-zero length, include a string terminator and not contain characters considered invalid for filenames.

If the file system free space table loaded at startup fails verification, the FM application will not terminate. However, the FM application will not process commands that request the file system free space telemetry packet if a file system free space table has not been successfully loaded. Thereafter, if an attempt to load a new table fails verification, the FM application will continue to use the previous table.

Definition at line 1295 of file fm_events.h.

Referenced by FM_ValidateTable().

10.6 CFS File Manager Command Structures

Data Structures

- struct FM_HousekeepingCmd_t

  Housekeeping Request command packet structure.
- struct FM_NoopCmd_t

  No-Operation command packet structure.
- struct FM_ResetCmd_t

  Reset Counters command packet structure.
- struct FM_CopyFileCmd_t

  Copy File command packet structure.
- struct FM_MoveFileCmd_t

  Move File command packet structure.
- struct FM_RenameFileCmd_t

  Rename File command packet structure.
- struct FM_DeleteFileCmd_t

  Delete File command packet structure.
- struct FM_DeleteAllCmd_t

  Delete All command packet structure.
- struct FM_DecompressCmd_t

  Decompress File command packet structure.
- struct FM_ConcatCmd_t

  Concatenate Files command packet structure.
- struct FM_GetFileInfoCmd_t

  Get File Info command packet structure.
- struct FM_GetOpenFilesCmd_t

  Get Open Files command packet structure.
- struct FM_CreateDirCmd_t

  Create Directory command packet structure.
- struct FM_DeleteDirCmd_t

  Delete Directory command packet structure.
- struct FM_GetDirFileCmd_t

  Get DIR List to File command packet structure.
- struct FM_GetDirPktCmd_t

  Get DIR List to Packet command packet structure.
- struct FM_GetFreeSpaceCmd_t

  Get Free Space command packet structure.
- struct FM_SetTableStateCmd_t

  Set Table State command packet structure.
- struct FM_SetPermCmd_t

  Set Permissions for a file.

10.6.1 Detailed Description

10.7 CFS File Manager Telemetry

Data Structures

- struct FM_DirListEntry_t

  Get Directory Listing entry structure.
- struct FM_DirListPkt_t

  Get Directory Listing telemetry packet.
- struct FM_DirListFileStats_t

  Get Directory Listing file statistics structure.
- struct FM_FileInfoPkt_t

  Get File Info telemetry packet.
- struct FM_OpenFilesEntry_t

  Get Open Files list entry structure.
- struct FM_OpenFilesPkt_t

  Get Open Files telemetry packet.
- struct FM_FreeSpacePktEntry_t

  Get Free Space list entry structure.
- struct FM_FreeSpacePkt_t

  Get Free Space telemetry packet.
- struct FM_HousekeepingPkt_t

  Housekeeping telemetry packet.

10.7.1 Detailed Description

10.8 CFS File Manager Command Codes

Macros

- #define FM_NOOP_CC 0

  No Operation.
- #define FM_RESET_CC 1

  Reset Counters.
- #define FM_COPY_CC 2

  Copy File.
- #define FM_MOVE_CC 3

  Move File.
- #define FM_RENAME_CC 4

  Rename File.
- #define FM_DELETE_CC 5

  Delete File.
- #define FM_DELETE_ALL_CC 7

  Delete All Files.
- #define FM_DECOMPRESS_CC 8

  Decompress File.
- #define FM_CONCAT_CC 9

  Concatenate Files.
- #define FM_GET_FILE_INFO_CC 10

  Get File Information.
- #define FM_GET_OPEN_FILES_CC 11

  Get Open Files Listing.
- #define FM_CREATE_DIR_CC 12

  Create Directory.
- #define FM_DELETE_DIR_CC 13

  Remove Directory.
- #define FM_GET_DIR_FILE_CC 14

  Get Directory Listing to a File.
- #define FM_GET_DIR_PKT_CC 15

  Get Directory Listing to a Packet.
- #define FM_GET_FREE_SPACE_CC 16

  Get Free Space.
- #define FM_SET_TABLE_STATE_CC 17

  Set Free Space Table Entry State.
- #define FM_DELETE_INT_CC 18

  Delete File (internal)
- #define FM_SET_FILE_PERM_CC 19

  Set Permissions of a file.

10.8.1 Detailed Description

10.8.2 Macro Definition Documentation

10.8.2.1 FM_CONCAT_CC

```
#define FM_CONCAT_CC 9
```

Concatenate Files.

Description

This command concatenates two source files into the target file. Sources must both be existing files and target must not exist. Sources and target may be on different file systems.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the first source file to the target file and then appending the second source file to the target file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_ConcatCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_CONCAT_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Invalid source filename
- Source file does not exist
- Invalid target filename
- Target file does exist
- Failure of OS function (copy, open, read, write, etc.)

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_CONCAT_PKT_ERR_EID may be sent
- Error event FM_CONCAT_OSCPY_ERR_EID may be sent
- Error event FM_CONCAT_OPEN_SRC2_ERR_EID may be sent
- Error event FM_CONCAT_OPEN_TGT_ERR_EID may be sent
- Error event FM_CONCAT_OSRD_ERR_EID may be sent
- Error event FM_CONCAT_OSWR_ERR_EID may be sent
- Error event FM_CONCAT_SRC1_INVALID_ERR_EID may be sent
- Error event FM_CONCAT_SRC1_DNE_ERR_EID may be sent
- Error event FM_CONCAT_SRC1_ISDIR_ERR_EID may be sent

- Error event FM_CONCAT_SRC1_OPEN_ERR_EID
- Error event FM_CONCAT_SRC2_INVALID_ERR_EID may be sent
- Error event FM_CONCAT_SRC2_DNE_ERR_EID may be sent
- Error event FM_CONCAT_SRC2_ISDIR_ERR_EID may be sent
- Error event FM_CONCAT_SRC2_OPEN_ERR_EID may be sent
- Error event FM_CONCAT_TGT_INVALID_ERR_EID may be sent
- Error event FM_CONCAT_TGT_EXIST_ERR_EID may be sent
- Error event FM_CONCAT_TGT_ISDIR_ERR_EID may be sent
- Error event FM_CONCAT_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_CONCAT_CHILD_FULL_ERR_EID may be sent
- Error event FM_CONCAT_CHILD_BROKEN_ERR_EID may be sent

Criticality

Concatenating very large files may consume more CPU resource than anticipated.

See also

FM_COPY_CC

Definition at line 522 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_ConcatFilesCmd(), and FM_ProcessCmd().

10.8.2.2 FM_COPY_CC

```
#define FM_COPY_CC 2
```

Copy File.

Description

This command copies the source file to the target file. The source must be an existing file and the target must not be a directory name. If the Overwrite command argument is TRUE, then the target may be an existing file, provided that the file is closed. If the Overwrite command argument is FALSE, then the target must not exist. The source and target may be on different file systems.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the file will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_CopyFileCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_COPY_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Overwrite is not TRUE (one) or FALSE (zero)
- Source filename is invalid
- Source file does not exist
- Source filename is a directory
- Target filename is invalid
- Target file already exists
- Target filename is a directory
- Child task interface queue is full
- Child task interface logic is broken
- Failure of OS copy function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_COPY_PKT_ERR_EID may be sent
- Error event FM_COPY_OVR_ERR_EID may be sent
- Error event FM_COPY_SRC_INVALID_ERR_EID may be sent
- Error event FM_COPY_SRC_DNE_ERR_EID may be sent
- Error event FM_COPY_SRC_ISDIR_ERR_EID may be sent
- Error event FM_COPY_SRC_UNKNOWN_ERR_EID may be sent
- Error event FM_COPY_TGT_INVALID_ERR_EID may be sent
- Error event FM_COPY_TGT_EXIST_ERR_EID may be sent
- Error event FM_COPY_TGT_ISDIR_ERR_EID may be sent
- Error event FM_COPY_TGT_UNKNOWN_ERR_EID may be sent
- Error event FM_COPY_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_COPY_CHILD_FULL_ERR_EID may be sent
- Error event FM_COPY_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_COPY_OS_ERR_EID may be sent

Criticality

Copying files may consume file space needed by other critical tasks. Also, copying very large files may consume more CPU resource than anticipated.

See also

FM_MOVE_CC, FM_RENAME_CC

Definition at line 158 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_CopyFileCmd(), and FM_ProcessCmd().

10.8.2.3 FM_CREATE_DIR_CC

```
#define FM_CREATE_DIR_CC 12
```

Create Directory.

Description

This command creates the source directory. Source must be a valid directory name that does not exist.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but creation of the directory will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_CreateDirCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_CREATE_DIR_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Invalid directory name
- Directory name already exists
- Failure of OS_mkdir function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_CREATE_DIR_PKT_ERR_EID may be sent
- Error event FM_CREATE_DIR_SRC_INVALID_ERR_EID may be sent
- Error event FM_CREATE_DIR_SRC_DNE_ERR_EID may be sent
- Error event FM_CREATE_DIR_SRC_ISDIR_ERR_EID may be sent
- Error event FM_CREATE_DIR_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_CREATE_DIR_CHILD_FULL_ERR_EID may be sent
- Error event FM_CREATE_DIR_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_CREATE_DIR_OS_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

See also

FM_DELETE_DIR_CC

Definition at line 661 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_CreateDirectoryCmd(), and FM_ProcessCmd().

10.8.2.4 FM_DECOMPRESS_CC

```
#define FM_DECOMPRESS_CC 8
```

Decompress File.

Description

This command invokes a CFE function to decompress the source file into the target file. Source must be an existing file and target must not exist. Source and target may be on different file systems.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but decompressing the source file into the target file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

This command is only valid if FM_INCLUDE_DECOMPRESS is defined.

Command Packet Structure

FM_DecompressCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_DECOM_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Invalid source filename
- Source file does not exist
- Invalid target filename
- Target file does exist
- Failure of CFE_FS_Decompress function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_DECOM_PKT_ERR_EID may be sent
- Error event FM_DECOM_SRC_INVALID_ERR_EID may be sent
- Error event FM_DECOM_SRC_DNE_ERR_EID may be sent
- Error event FM_DECOM_SRC_ISDIR_ERR_EID may be sent
- Error event FM_DECOM_SRC_OPEN_ERR_EID may be sent
- Error event FM_DECOM_TGT_INVALID_ERR_EID may be sent
- Error event FM_DECOM_TGT_EXIST_ERR_EID may be sent

- Error event FM_DECOM_TGT_ISDIR_ERR_EID may be sent
- Error event FM_DECOM_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_DECOM_CHILD_FULL_ERR_EID may be sent
- Error event FM_DECOM_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_DECOM_CFE_ERR_EID may be sent

Criticality

Decompressing a very large file may consume more CPU resource than anticipated.

Definition at line 456 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_DeleteAllFilesCmd(), and FM_ProcessCmd().

10.8.2.5 FM_DELETE_ALL_CC

```
#define FM_DELETE_ALL_CC 7
```

Delete All Files.

Description

This command deletes all files in the source directory. Source must be an existing directory. Open files and sub-directories are not deleted.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory and deleting each file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_DeleteAllCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_DELETE_ALL_CMD_EID will be sent

Command Warning Conditions

- Directory entry is not a file (sub-directory)
- Directory entry is an open file

Command Warning Verification

- FM_HousekeepingPkt_t.ChildCmdWarnCounter will increment
- Informational event FM_DELETE_ALL_FILES_ND_WARNING_EID may be sent
- Informational event FM_DELETE_ALL_SKIP_WARNING_EID may be sent

Command Error Conditions

- Invalid command packet length
- Invalid directory name
- Directory does not exist
- Directory name + separator + filename is too long
- Failure of OS delete function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_DELETE_ALL_PKT_ERR_EID may be sent
- Error event FM_DELETE_ALL_SRC_INVALID_ERR_EID may be sent
- Error event FM_DELETE_ALL_SRC_DNE_ERR_EID may be sent
- Error event FM_DELETE_ALL_SRC_FILE_ERR_EID may be sent
- Error event FM_DELETE_ALL_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_DELETE_ALL_CHILD_FULL_ERR_EID may be sent
- Error event FM_DELETE_ALL_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_DELETE_ALL_OS_ERR_EID may be sent

Criticality

The FM application does not provide a method to restore deleted files. Critical data may be lost when deleting files. Also, deleting a very large number of files may consume more CPU resource than anticipated.

See also

FM_DELETE_CC, FM_DELETE_DIR_CC

Definition at line 399 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_DeleteAllFilesCmd(), and FM_ProcessCmd().

10.8.2.6 FM_DELETE_CC

```
#define FM_DELETE_CC 5
```

Delete File.

Description

This command deletes the source file. Source must be an existing file that is not open.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but deleting the file will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_DeleteFileCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_DELETE_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Filename is invalid
- File does not exist
- File is open
- Filename is a directory
- Failure of OS delete function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_DELETE_PKT_ERR_EID may be sent
- Error event FM_DELETE_SRC_INVALID_ERR_EID may be sent
- Error event FM_DELETE_SRC_DNE_ERR_EID may be sent
- Error event FM_DELETE_SRC_ISDIR_ERR_EID may be sent
- Error event FM_DELETE_SRC_OPEN_ERR_EID may be sent
- Error event FM_DELETE_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_DELETE_CHILD_FULL_ERR_EID may be sent
- Error event FM_DELETE_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_DELETE_OS_ERR_EID may be sent

Criticality

The FM application does not provide a method to restore deleted files. Critical data may be lost when deleting files.

See also

FM_DELETE_ALL_CC, FM_DELETE_DIR_CC

Definition at line 337 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), and FM_ProcessCmd().

10.8.2.7 FM_DELETE_DIR_CC

```
#define FM_DELETE_DIR_CC 13
```

Remove Directory.

Description

This command deletes the source directory, it does not delete the contents of the directory. Source must be a valid directory name that exists.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but removal of the directory will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_DeleteDirCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_DELETE_DIR_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Invalid directory name
- Directory does not exist
- Directory is not empty
- Failure of OS function (OS_opendir, OS_rmdir)

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_DELETE_DIR_PKT_ERR_EID may be sent
- Error event FM_DELETE_DIR_EMPTY_ERR_EID may be sent

- Error event FM_DELETE_OPENDIR_OS_ERR_EID may be sent
- Error event FM_DELETE_RMDIR_OS_ERR_EID may be sent
- Error event FM_DELETE_DIR_SRC_INVALID_ERR_EID may be sent
- Error event FM_DELETE_DIR_SRC_DNE_ERR_EID may be sent
- Error event FM_DELETE_DIR_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_DELETE_DIR_CHILD_FULL_ERR_EID may be sent
- Error event FM_DELETE_DIR_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_DELETE_RMDIR_OS_ERR_EID may be sent

Criticality

The unexpected loss of a directory may affect a critical tasks ability to store data.

See also

FM_CREATE_DIR_CC

Definition at line 713 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_DeleteDirectoryCmd(), and FM_ProcessCmd().

10.8.2.8 FM_DELETE_INT_CC

```
#define FM_DELETE_INT_CC 18
```

Delete File (internal)

Description

This is a special version of the FM_DELETE_CC command for use when the command is sent by another application, rather than from the ground. This version of the command will not generate a success event, nor will the command increment the command success counter. The intent is to avoid confusion resulting from telemetry representing the results of delete commands sent by other applications and those sent from the ground. Refer to FM_DELETE_CC command for use details.

See also

FM_DELETE_CC

Definition at line 940 of file fm_msgdefs.h.

Referenced by FM_ChildDeleteCmd(), FM_ChildProcess(), and FM_ProcessCmd().

10.8.2.9 FM_GET_DIR_FILE_CC

```
#define FM_GET_DIR_FILE_CC 14
```

Get Directory Listing to a File.

Description

This command writes a listing of the contents of the source directory to the target file. If the target filename buffer is empty, then the default target filename FM_DIR_LIST_FILE_DEFNAME is used. The command will overwrite a previous copy of the target file, if one exists.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_GetDirFileCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_GET_DIR_FILE_CMD_EID will be sent

Command Warning Conditions

- Combined directory and entry name is too long

Command Warning Verification

- FM_HousekeepingPkt_t.ChildCmdWarnCounter will increment
- Informational event FM_GET_DIR_FILE_WARNING_EID may be sent

Command Error Conditions

- Invalid command packet length
- Invalid source directory name
- Source directory does not exist
- Directory name + separator is too long
- Directory name + directory entry is too long
- Invalid target filename
- Target file is already open
- Failure of OS function (OS_opendir, OS_creat, OS_write)

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment

- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_GET_DIR_FILE_PKT_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_OSOPENDIR_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_WRBLANK_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_WRHDR_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_OSCREAT_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_WRENTRY_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_UPSTATS_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_SRC_INVALID_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_SRC_DNE_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_TGT_INVALID_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_CHILD_FULL_ERR_EID may be sent
- Error event FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID may be sent

Criticality

Reading a directory that contains thousands of files may consume more CPU resource than anticipated.

See also

FM_GET_DIR_PKT_CC

Definition at line 783 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_GetDirListFileCmd(), and FM_ProcessCmd().

10.8.2.10 FM_GET_DIR_PKT_CC

```
#define FM_GET_DIR_PKT_CC 15
```

Get Directory Listing to a Packet.

Description

This command creates a telemetry packet FM_DirListPkt_t that contains a listing of the entries in the specified directory. Since the packet will likely hold fewer entries than will be possible in a directory, the command also provides an index argument to define which entry in the directory is the first entry reported in the telemetry packet. After reading the directory list and skipping entries until reaching the index of the first entry reported, the remaining entries in the packet are filled sequentially until either the packet is full or until there are no more entries in the directory. The first entry index is zero based - thus, when the first entry index is zero the first directory entry will be the first packet entry. The number of entries per packet FM_DIR_LIST_PKT_ENTRIES is a platform configuration definition.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_GetDirPktCmd_t

Command Verification

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

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- The FM_DirListPkt_t telemetry packet will be sent
- The FM_GET_DIR_PKT_CMD_EID debug event will be sent

Command Warning Conditions

- Combined directory and entry name is too long

Command Warning Verification

- FM_HousekeepingPkt_t.ChildCmdWarnCounter will increment
- Informational event FM_GET_DIR_PKT_WARNING_EID may be sent

Error Conditions

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

- OS error received opening directory
- OS error received requesting directory size
- OS error received closing directory
- Invalid directory pathname received
- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_GET_DIR_PKT_PKT_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_OS_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_SRC_INVALID_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_SRC_DNE_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_CHILD_FULL_ERR_EID may be sent
- Error event FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID may be sent

Criticality

Reading a directory that contains thousands of files may consume more CPU resource than anticipated.

See also

FM_GET_DIR_FILE_CC

Definition at line 857 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_GetDirListPktCmd(), and FM_ProcessCmd().

10.8.2.11 FM_GET_FILE_INFO_CC

```
#define FM_GET_FILE_INFO_CC 10
```

Get File Information.

Description

This command creates an FM file information telemetry packet for the source file. The file information packet includes status that indicates whether source is a file that is open or closed, a directory, or does not exist. The file information data also includes a CRC, file size, last modify time and the source name.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but collecting the status data and calculating the CRC will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_GetFileInfoCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_GET_FILE_INFO_CMD_EID will be sent

Command Warning Conditions

- File is open and CRC cannot be calculated
- Specified CRC type is not valid
- CRC cannot be calculated because file cannot be read

Command Warning Verification

- FM_HousekeepingPkt_t.ChildCmdWarnCounter will increment
- Informational event FM_GET_FILE_INFO_STATE_WARNING_EID may be sent
- Informational event FM_GET_FILE_INFO_TYPE_WARNING_EID may be sent
- Informational event FM_GET_FILE_INFO_READ_WARNING_EID may be sent

Command Error Conditions

- Invalid command packet length
- Invalid source filename
- Failure of OS_stat function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_GET_FILE_INFO_OPEN_ERR_EID may be sent
- Error event FM_GET_FILE_INFO_PKT_ERR_EID may be sent
- Error event FM_GET_FILE_INFO_SRC_ERR_EID may be sent
- Error event FM_FILE_INFO_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_FILE_INFO_CHILD_FULL_ERR_EID may be sent
- Error event FM_FILE_INFO_CHILD_BROKEN_ERR_EID may be sent

Criticality

Calculating the CRC for a very large file may consume more CPU resource than anticipated.

See also

FM_GET_OPEN_FILES_CC, FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC

Definition at line 584 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_GetFileInfoCmd(), and FM_ProcessCmd().

10.8.2.12 FM_GET_FREE_SPACE_CC

```
#define FM_GET_FREE_SPACE_CC 16
```

Get Free Space.

Description

This command queries the amount of free space for each of the enabled entries in the file system free space table. The data is then placed in a telemetry packet and sent to ground.

Command Packet Structure

FM_GetFreeSpaceCmd_t

Evidence of success may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandCounter will increment

- Debug event FM_GET_FREE_SPACE_CMD_EID will be sent
- Telemetry packet FM_FreeSpacePkt_t will be sent

Error Conditions

- Invalid command packet length
- Free space table is not loaded

Evidence of failure may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- Error event FM_GET_FREE_SPACE_PKT_ERR_EID may be sent
- Error event FM_GET_FREE_SPACE_TBL_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

Definition at line 887 of file fm_msgdefs.h.

Referenced by FM_ProcessCmd().

10.8.2.13 FM_GET_OPEN_FILES_CC

```
#define FM_GET_OPEN_FILES_CC 11
```

Get Open Files Listing.

Description

This command creates an FM open files telemetry packet. The open files packet includes the number of open files and for each open file, the name of the file and the name of the application that has the file opened.

Command Packet Structure

FM_GetOpenFilesCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment
- Debug event FM_GET_OPEN_FILES_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- Error event FM_GET_OPEN_FILES_PKT_ERR_EID will be sent

Criticality

- There are no critical issues related to this command.

See also

FM_GET_FILE_INFO_CC, FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC

Definition at line 614 of file fm_msgdefs.h.

Referenced by FM_ProcessCmd().

10.8.2.14 FM_MOVE_CC

```
#define FM_MOVE_CC 3
```

Move File.

Description

This command moves the source file to the target file. The source must be an existing file and the target must not be a directory name. If the Overwrite command argument is TRUE, then the target may be an existing file, provided that the file is closed. If the Overwrite command argument is FALSE, then the target must not exist. Source and target must both be on the same file system. The move command does not actually move any file data. The command modifies the file system directory structure to create a different file entry for the same file data. If the user wishes to move a file across file systems, he must first copy the file and then delete the original.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but moving the file will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_MoveFileCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_MOVE_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Overwrite is not TRUE (one) or FALSE (zero)
- Source filename is invalid
- Source file does not exist
- Source filename is a directory
- Target filename is invalid
- Target file already exists
- Target filename is a directory
- Failure of OS move function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_MOVE_PKT_ERR_EID may be sent
- Error event FM_MOVE_OVR_ERR_EID may be sent
- Error event FM_MOVE_SRC_INVALID_ERR_EID may be sent

- Error event FM_MOVE_SRC_DNE_ERR_EID may be sent
- Error event FM_MOVE_SRC_ISDIR_ERR_EID may be sent
- Error event FM_MOVE_TGT_INVALID_ERR_EID may be sent
- Error event FM_MOVE_TGT_EXIST_ERR_EID may be sent
- Error event FM_MOVE_TGT_ISDIR_ERR_EID may be sent
- Error event FM_MOVE_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_MOVE_CHILD_FULL_ERR_EID may be sent
- Error event FM_MOVE_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_MOVE_OS_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

See also

FM_COPY_CC, FM_RENAME_CC

Definition at line 226 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_MoveFileCmd(), and FM_ProcessCmd().

10.8.2.15 FM_NOOP_CC

```
#define FM_NOOP_CC 0
```

No Operation.

Description

This command performs no operation other than to generate an informational event that also contains software version data. The command is most often used as a general aliveness test by demonstrating that the application can receive commands and generate telemetry.

Command Packet Structure

FM_NoopCmd_t

Command Success Verification

- Informational event FM_NOOP_CMD_EID will be sent
- FM_HousekeepingPkt_t.CommandCounter will increment

Command Error Conditions

- Invalid command packet length

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- Error event FM_NOOP_PKT_ERR_EID will be sent

Criticality

- There are no critical issues related to this command.

Definition at line 60 of file fm_msgdefs.h.

Referenced by FM_ProcessCmd().

10.8.2.16 FM_RENAME_CC

```
#define FM_RENAME_CC 4
```

Rename File.

Description

This command renames the source file to the target file. Source must be an existing file and target must not exist. Source and target must both be on the same file system. The rename command does not actually move any file data. The command modifies the file system directory structure to create a different file entry for the same file data. If the user wishes to rename a file across file systems, he must first copy the file and then delete the original.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but renaming the file will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_RenameFileCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_RENAME_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length
- Source filename is invalid
- Source file does not exist
- Source filename is a directory
- Target filename is invalid
- Target file already exists
- Target filename is a directory
- Failure of OS rename function

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_RENAME_PKT_ERR_EID may be sent
- Error event FM_RENAME_SRC_INVALID_ERR_EID may be sent
- Error event FM_RENAME_SRC_DNE_ERR_EID may be sent
- Error event FM_RENAME_SRC_ISDIR_ERR_EID may be sent
- Error event FM_RENAME_TGT_INVALID_ERR_EID may be sent

- Error event FM_RENAME_TGT_EXIST_ERR_EID may be sent
- Error event FM_RENAME_TGT_ISDIR_ERR_EID may be sent
- Error event FM_RENAME_CHILD_DISABLED_ERR_EID may be sent
- Error event FM_RENAME_CHILD_FULL_ERR_EID may be sent
- Error event FM_RENAME_CHILD_BROKEN_ERR_EID may be sent
- Error event FM_RENAME_OS_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

See also

FM_COPY_CC, FM_MOVE_CC

Definition at line 286 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_ProcessCmd(), and FM_RenameFileCmd().

10.8.2.17 FM_RESET_CC

```
#define FM_RESET_CC 1
```

Reset Counters.

Description

This command resets the following housekeeping telemetry:

- FM_HousekeepingPkt_t.CommandCounter
- FM_HousekeepingPkt_t.CommandErrCounter
- FM_HousekeepingPkt_t.ChildCmdCounter
- FM_HousekeepingPkt_t.ChildCmdErrCounter
- FM_HousekeepingPkt_t.ChildCmdWarnCounter

Command Packet Structure

FM_ResetCmd_t

Command Success Verification

- Command counters will be set to zero (see description)
- Debug event FM_RESET_CMD_EID will be sent

Command Error Conditions

- Invalid command packet length

Command Failure Verification

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- Error event FM_RESET_PKT_ERR_EID will be sent

Criticality

- There are no critical issues related to this command.

Definition at line 91 of file fm_msgdefs.h.

Referenced by FM_ProcessCmd().

10.8.2.18 FM_SET_FILE_PERM_CC

```
#define FM_SET_FILE_PERM_CC 19
```

Set Permissions of a file.

Description

This command sets the permissions for a file. This is a direct interface to OS_chmod in the OSAL. OS_chmod accepts a uint32 to set the file's mode.

Examples for a regular file:

OS_READ_ONLY - Read only file access OS_WRITE_ONLY - Write only file access OS_READ_WRITE - Read write file access

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but setting permissions will be performed by a lower priority child task. As such, the command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Packet Structure

FM_SetPermCmd_t

Command Success Verification

- FM_HousekeepingPkt_t.CommandCounter will increment after validation
- FM_HousekeepingPkt_t.ChildCmdCounter will increment after completion
- Debug event FM_SET_PERM_CMD_EID will be sent

Error Conditions

- Invalid command packet length
- Error from call to OS_chmod

Evidence of failure may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandErrCounter may increment
- FM_HousekeepingPkt_t.ChildCmdErrCounter may increment
- Error event FM_SET_PERM_ERR_EID may be sent
- Error event FM_SET_PERM_OS_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

Definition at line 983 of file fm_msgdefs.h.

Referenced by FM_ChildProcess(), FM_ProcessCmd(), and FM_SetPermissionsCmd().

10.8.2.19 FM_SET_TABLE_STATE_CC

```
#define FM_SET_TABLE_STATE_CC 17
```

Set Free Space Table Entry State.

Description

This command enables or disables a single entry in the FM file system free space table. Only table entries that are currently enabled or disabled may be modified by command. Unused table entries cannot be modified.

Command Packet Structure

FM_SetTableStateCmd_t

Evidence of success may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandCounter will increment
- Informational event FM_SET_TABLE_STATE_CMD_EID will be sent

Error Conditions

- Invalid command packet length
- FM file system free space table has not yet been loaded
- Invalid command argument, table entry index arg is out of range
- Invalid command argument, entry state arg is not enable/disable
- Invalid current table entry state, entry is unused

Evidence of failure may be found in the following telemetry:

- FM_HousekeepingPkt_t.CommandErrCounter will increment
- Error event FM_SET_TABLE_STATE_PKT_ERR_EID may be sent
- Error event FM_SET_TABLE_STATE_TBL_ERR_EID may be sent
- Error event FM_SET_TABLE_STATE_ARG_IDX_ERR_EID may be sent
- Error event FM_SET_TABLE_STATE_ARG_STATE_ERR_EID may be sent
- Error event FM_SET_TABLE_STATE_UNUSED_ERR_EID may be sent

Criticality

- There are no critical issues related to this command.

Definition at line 923 of file fm_msgdefs.h.

Referenced by FM_ProcessCmd().

10.9 CFS File Manager Version

Macros

- #define FM_MAJOR_VERSION 2

  Major version number.
- #define FM_MINOR_VERSION 6

  Minor version number.
- #define FM_REVISION 99

  Revision number.

10.9.1 Detailed Description

Version Numbers

10.9.2 Macro Definition Documentation

10.9.2.1 FM_MAJOR_VERSION

```
#define FM_MAJOR_VERSION 2
```

Major version number.

Definition at line 35 of file fm_version.h.

Referenced by FM_AppInit(), and FM_NoopCmd().

10.9.2.2 FM_MINOR_VERSION

```
#define FM_MINOR_VERSION 6
```

Minor version number.

Definition at line 36 of file fm_version.h.

Referenced by FM_AppInit(), and FM_NoopCmd().

10.9.2.3 FM_REVISION

```
#define FM_REVISION 99
```

Revision number.

Definition at line 37 of file fm_version.h.

Referenced by FM_AppInit(), and FM_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)

Max Messages Met.

- #define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca00000a)

  Max Destinations Met.
- #define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca00000c)

  Internal Error.
- #define CFE_SB_WRONG_MSG_TYPE ((CFE_Status_t)0xca00000d)

  Wrong Message Type.
- #define CFE_SB_BUFFER_INVALID ((CFE_Status_t)0xca00000e)

  Buffer Invalid.
- #define CFE_SB_NOT_IMPLEMENTED ((CFE_Status_t)0xca00ffff)

  Not Implemented.
- #define CFE_TBL_ERR_INVALID_HANDLE ((CFE_Status_t)0xcc000001)

  Invalid Handle.
- #define CFE_TBL_ERR_INVALID_NAME ((CFE_Status_t)0xcc000002)

  Invalid Name.
- #define CFE_TBL_ERR_INVALID_SIZE ((CFE_Status_t)0xcc000003)

  Invalid Size.
- #define CFE_TBL_INFO_UPDATE_PENDING ((CFE_Status_t)0x4c000004)

  Update Pending.
- #define CFE_TBL_ERR_NEVER_LOADED ((CFE_Status_t)0xcc000005)

  Never Loaded.
- #define CFE_TBL_ERR_REGISTRY_FULL ((CFE_Status_t)0xcc000006)

  Registry Full.
- #define CFE_TBL_WARN_DUPLICATE ((CFE_Status_t)0x4c000007)

  Duplicate Warning.
- #define CFE_TBL_ERR_NO_ACCESS ((CFE_Status_t)0xcc000008)

  No Access.
- #define CFE_TBL_ERR_UNREGISTERED ((CFE_Status_t)0xcc000009)

  Unregistered.
- #define CFE_TBL_ERR_HANDLES_FULL ((CFE_Status_t)0xcc00000B)

  Handles Full.
- #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((CFE_Status_t)0xcc00000C)

  Duplicate Table With Different Size.
- #define CFE_TBL_ERR_DUPLICATE_NOT_OWNED ((CFE_Status_t)0xcc00000D)

  Duplicate Table And Not Owned.
- #define CFE_TBL_INFO_UPDATED ((CFE_Status_t)0x4c00000E)

  Updated.
- #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((CFE_Status_t)0xcc00000F)

  No Buffer Available.
- #define CFE_TBL_ERR_DUMP_ONLY ((CFE_Status_t)0xcc000010)

  Dump Only Error.
- #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((CFE_Status_t)0xcc000011)

  Illegal Source Type.
- #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((CFE_Status_t)0xcc000012)

  Load In Progress.
- #define CFE_TBL_ERR_FILE_TOO_LARGE ((CFE_Status_t)0xcc000014)

  File Too Large.

- #define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)

  Short File Warning.
- #define CFE_TBL_ERR_BAD_CONTENT_ID ((CFE_Status_t)0xcc000016)

  Bad Content ID.
- #define CFE_TBL_INFO_NO_UPDATE_PENDING ((CFE_Status_t)0x4c000017)

  No Update Pending.
- #define CFE_TBL_INFO_TABLE_LOCKED ((CFE_Status_t)0x4c000018)

  Table Locked.
- #define CFE_TBL_INFO_VALIDATION_PENDING ((CFE_Status_t)0x4c000019)
- #define CFE_TBL_INFO_NO_VALIDATION_PENDING ((CFE_Status_t)0x4c00001A)
- #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((CFE_Status_t)0xcc00001B)

  Bad Subtype ID.
- #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((CFE_Status_t)0xcc00001C)

  File Size Inconsistent.
- #define CFE_TBL_ERR_NO_STD_HEADER ((CFE_Status_t)0xcc00001D)

  No Standard Header.
- #define CFE_TBL_ERR_NO_TBL_HEADER ((CFE_Status_t)0xcc00001E)

  No Table Header.
- #define CFE_TBL_ERR_FILENAME_TOO_LONG ((CFE_Status_t)0xcc00001F)

  Filename Too Long.
- #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc000020)

  File For Wrong Table.
- #define CFE_TBL_ERR_LOAD_INCOMPLETE ((CFE_Status_t)0xcc000021)

  Load Incomplete.
- #define CFE_TBL_WARN_PARTIAL_LOAD ((CFE_Status_t)0x4c000022)

  Partial Load Warning.
- #define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)

  Partial Load Error.
- #define CFE_TBL_INFO_DUMP_PENDING ((CFE_Status_t)0x4c000024)

  Dump Pending.
- #define CFE_TBL_ERR_INVALID_OPTIONS ((CFE_Status_t)0xcc000025)

  Invalid Options.
- #define CFE_TBL_WARN_NOT_CRITICAL ((CFE_Status_t)0x4c000026)

  Not Critical Warning.
- #define CFE_TBL_INFO_RECOVERED_TBL ((CFE_Status_t)0x4c000027)

  Recovered Table.
- #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((CFE_Status_t)0xcc000028)

  Bad Spacecraft ID.
- #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((CFE_Status_t)0xcc000029)

  Bad Processor ID.
- #define CFE_TBL_MESSAGE_ERROR ((CFE_Status_t)0xcc00002a)

  Message Error.
- #define CFE_TBL_ERR_SHORT_FILE ((CFE_Status_t)0xcc00002b)
- #define CFE_TBL_ERR_ACCESS ((CFE_Status_t)0xcc00002c)
- #define CFE_TBL_BAD_ARGUMENT ((CFE_Status_t)0xcc00002d)

  Bad Argument.
- #define CFE_TBL_NOT_IMPLEMENTED ((CFE_Status_t)0xcc00ffff)

Not Implemented.

- #define CFE_TIME_NOT_IMPLEMENTED ((CFE_Status_t)0xce00ffff)

  Not Implemented.

- #define CFE_TIME_INTERNAL_ONLY ((CFE_Status_t)0xce000001)

  Internal Only.

- #define CFE_TIME_OUT_OF_RANGE ((CFE_Status_t)0xce000002)

  Out Of Range.

- #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((CFE_Status_t)0xce000003)

  Too Many Sync Callbacks.

- #define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000004)

  Callback Not Registered.

- #define CFE_TIME_BAD_ARGUMENT ((CFE_Status_t)0xce000005)

  Bad Argument.

10.10.1 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.

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_MI↩SSION_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_FIL↩TERS

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)0xca00000e)
```

Buffer Invalid.

This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Definition at line 798 of file cfe_error.h.

10.10.2.56 CFE_SB_INTERNAL_ERR

```
#define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca00000c)
```

Internal Error.

This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Definition at line 779 of file cfe_error.h.

10.10.2.57 CFE_SB_MAX_DESTS_MET

```
#define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca00000a)
```

Max Destinations Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.

Definition at line 769 of file cfe_error.h.

10.10.2.58 CFE_SB_MAX_MSGS_MET

```
#define CFE_SB_MAX_MSGS_MET ((CFE_Status_t)0xca000009)
```

Max Messages Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.

Definition at line 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.

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 FM_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 FM_AppInit(), FM_AppMain(), FM_ChildDeleteAllCmd(), FM_ChildInit(), FM_ChildLoop(), FM_Table↩Init(), and FM_ValidateTable().

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.

Referenced by FM_AcquireTablePointers().

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_N↩UM_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.

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	*AppID*	Application ID to convert
out	*Idx*	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.

10.11.2.2 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	*CounterId*	Counter ID to convert
out	*Idx*	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.

10.11.2.3 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	*LibId*	Library ID to convert
out	*Idx*	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	*TaskID*	Task ID to convert
out	*Idx*	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.
CFE_ES_BAD_ARGUMENT	Bad Argument.

10.12 cFE Entry/Exit APIs

Functions

- void CFE_ES_Main (uint32 StartType, uint32 StartSubtype, uint32 ModeId, const char *StartFilePath)

  cFE Main Entry Point used by Board Support Package to start cFE
- 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 ModeId,
            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	*StartType*	Identifies whether this was a CFE_PSP_RST_TYPE_POWERON or CFE_PSP_RST_TYPE_PROCESSOR.
in	*StartSubtype*	Specifies, in more detail, what caused the `StartType` identified above. See CFE_PSP_RST_SUBTYPE_POWER_CYCLE for possible examples.
in	*ModeId*	Identifies the source of the Boot as determined by the BSP.
in	*StartFilePath*	Identifies the startup file to use to initialize the cFE apps.

See also

CFE_ES_ResetCFE

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	*ResetType*	Identifies the type of reset desired. Allowable settings are: • CFE_PSP_RST_TYPE_POWERON - Causes all memory to be cleared • CFE_PSP_RST_TYPE_PROCESSOR - Attempts to retain volatile disk, critical data store and user reserved memory.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NOT_IMPLEMENTED	Not Implemented.

See also

CFE_ES_Main

10.13 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	*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_SUCCESS	Successful execution.

See also

CFE_ES_RestartApp, CFE_ES_ReloadApp

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	*AppID*	Identifies the application to be reset.
in	*AppFileName*	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

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	*ExitStatus*	Acceptable values are: • 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 FM_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

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_RES↩TART 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 FM_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	*TimeOutMilliseconds*	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

See also

CFE_ES_RunLoop

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	*MinSystemState*	Determine the state of the App
in	*TimeOutMilliseconds*	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_ResourceId_t ResourceId)

  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	*AppIdPtr*	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

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_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	*AppIdPtr*	Pointer to variable that is to receive the Application's ID (must not be null).
in	*AppName*	Pointer to null terminated character string containing an Application 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_GetAppID, CFE_ES_GetAppName, CFE_ES_GetAppInfo

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`	*AppInfo*	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
`in`	*AppId*	ID of application to obtain information about

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName

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	*AppName*	Pointer to a character array (must not be null) of at least `BufferLength` in size that will be filled with the appropriate Application name.
in	*AppId*	Application ID of Application whose name is being requested.
in	*BufferLength*	The maximum number of characters, including the null terminator, that can be put into the `AppName` buffer. This routine will truncate the name to this length, if necessary.

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

out	*LibIdPtr*	Pointer to variable that is to receive the Library's ID (must not be null).
in	*LibName*	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	*LibInfo*	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	*LibId*	ID of application to obtain information about

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_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

out	*LibName*	Pointer to a character array (must not be null) of at least `BufferLength` in size that will be filled with the Library name.
in	*LibId*	Library ID of Library whose name is being requested.
in	*BufferLength*	The maximum number of characters (must not be zero), including the null terminator, that can be put into the `LibName` 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_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	*ModuleInfo*	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	*ResourceId*	ID of application or library to obtain information about

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_GetLibInfo, CFE_ES_GetAppInfo

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	*ResetSubtypePtr*	Pointer to `uint32` type variable in which the Reset Sub-Type will be stored. The caller can set this pointer to NULL if the Sub-Type is of no interest. ResetSubtypePtr If the provided pointer was not `NULL`, the Reset Sub-Type is stored at the given address. For a list of possible Sub-Type values, see "Reset Sub-Types" .

Returns

Processor reset type

Return values

CFE_PSP_RST_TYPE_POWERON	
CFE_PSP_RST_TYPE_PROCESSOR	

See also

CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

10.15.2.10 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

out	*TaskIdPtr*	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

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	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	*TaskInfo*	Pointer to a `CFE_ES_TaskInfo_t` structure (must not be null) that holds the specific task information. *TaskInfo is the filled out `CFE_ES_TaskInfo_t` structure containing the Task Name, Parent App Name, Parent App ID among other fields.
in	*TaskId*	Application ID of Application whose name is being requested.

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_GetTaskID, CFE_ES_GetTaskIDByName, CFE_ES_GetTaskName

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	*TaskIdPtr*	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	*TaskName*	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	*FunctionPtr*	A pointer to the function that will be spawned as a new task (must not be null).
in	*StackPtr*	A pointer to the location where the child task's stack pointer should start. NOTE: Not all underlying operating systems support this parameter. The CFE_ES_TASK_STACK_ALLOCATE constant may be passed to indicate that the stack should be dynamically allocated.
in	*StackSize*	The number of bytes to allocate for the new task's stack (must not be zero).
in	*Priority*	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority.
in	*Flags*	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 FM_ChildInit().

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	*TaskId*	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

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

Referenced by FM_ChildTask().

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	*TaskIdPtr*	Pointer to variable that is to receive the Task's ID (must not be null).
in	*TaskName*	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

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`	*TaskName*	Pointer to a character array (must not be null) of at least `BufferLength` in size that will be filled with the Task name.
`in`	*TaskId*	Task ID of Task whose name is being requested.
`in`	*BufferLength*	The maximum number of characters, including the null terminator, that can be put into the `TaskName` 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	*DataPtr*	Pointer to the base of the memory block.
in	*DataLength*	The number of bytes in the memory block.
in	*InputCRC*	A starting value for use in the CRC calculation. This parameter allows the user to calculate the CRC of non-contiguous blocks as a single value. Nominally, the user should set this value to zero.
in	*TypeCRC*	One of the following CRC algorithm selections: • `CFE_MISSION_ES_CRC_8` - (Not currently implemented) • `CFE_MISSION_ES_CRC_16` - 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

Referenced by FM_ChildFileInfoCmd().

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	*SpecStringPtr*	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

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_SYS_LOG_FULL	System Log Full.
CFE_ES_BAD_ARGUMENT	Bad Argument.

Referenced by FM_AppMain().

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 Buffer↩ Length)

  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	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
in	*DataToCopy*	A Pointer to the block of memory to be copied into the CDS (must not be null).

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_RegisterCDS, CFE_ES_RestoreFromCDS

10.18.2.2 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

out	*BlockIdPtr*	Pointer to variable that is to receive the CDS Block ID (must not be null).
in	*BlockName*	Pointer to null terminated character string containing a CDS Block name (must not be null).

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_NAME_NOT_FOUND	Resource Name Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.

See also

CFE_ES_GetCDSBlockName

10.18.2.3 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	*BlockName*	Pointer to a character array (must not be null) of at least `BufferLength` in size that will be filled with the CDS Block name.
in	*BlockId*	Block ID/Handle of CDS registry entry whose name is being requested.
in	*BufferLength*	The maximum number of characters, including the null terminator, that can be put into the `BlockName` 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	*CDSHandlePtr*	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	*BlockSize*	The number of bytes needed in the CDS (must not be zero).
in	*Name*	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.
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

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`	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
`out`	*RestoreToMemory*	A Pointer to the block of memory (must not be null) that is to be restored with the contents of the CDS. ∗RestoreToMemory is the contents of the specified CDS.

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_CDS_BLOCK_CRC_ERR	(return value only verified in coverage test) CDS Block CRC Error.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_RegisterCDS, CFE_ES_CopyToCDS

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

out	*BufPtr*	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.
in	*Handle*	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

out	*BufPtr*	A pointer to the Application's pointer (must not be null) in which will be stored the address of the allocated memory buffer. ∗BufPtr is the address of the requested buffer.
in	*Handle*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*Size*	The size of the buffer requested. NOTE: The size allocated may be larger.

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	*Handle*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to provide status for (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	*PoolID*	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	*MemPtr*	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	*Size*	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	*PoolID*	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	*MemPtr*	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	*Size*	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	*NumBlockSizes*	The number of different block sizes specified in the `BlockSizes` 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	*BlockSizes*	Pointer to an array of sizes to be used instead of the default block sizes specified by CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 through CFE_PLATFORM_ES_MAX_BLOCK_SIZE. If the pointer is equal to NULL, the default block sizes are used.
in	*UseMutex*	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are CFE_ES_USE_MUTEX and CFE_ES_NO_MUTEX

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.
CFE_ES_NO_RESOURCE_IDS_AVAILABLE	Resource ID is not available.
CFE_STATUS_EXTERNAL_RESOURCE_FAIL	(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

out	*PoolID*	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	*MemPtr*	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	*Size*	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_Get↩MemPoolStats

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	*PoolID*	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	*Handle*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to be released (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	*id*	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 FM_AppMain(), FM_ChildConcatCmd(), FM_ChildFileInfoCmd(), FM_ChildLoop(), and FM_Child↩ SleepStat().

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	*id*	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 FM_AppMain(), FM_ChildConcatCmd(), FM_ChildFileInfoCmd(), FM_ChildLoop(), and FM_Child↩ SleepStat().

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`	*Marker*	Identifier of the specific event or marker.
`in`	*EntryExit*	Used to specify Entry(0) or Exit(1)

See also

CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit

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	*Counter↩Id*	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_GetGen↩Count, 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	*Counter↩Id*	The Counter to get the value from.
out	*Count*	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_Increment↩GenCounter, CFE_ES_GetGenCounterIDByName

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	*CounterIdPtr*	Pointer to variable that is to receive the Counter's ID (must not be null).
in	*CounterName*	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

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	*CounterName*	Pointer to a character array (must not be null) of at least `BufferLength` in size that will be filled with the Counter name.
in	*CounterId*	ID of Counter whose name is being requested.
in	*BufferLength*	The maximum number of characters, including the null terminator (must not be zero), that can be put into the `CounterName` 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	*CounterId*	The Counter to be incremented.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_BAD_ARGUMENT	Bad Argument.

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

10.21.2.6 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	*CounterIdPtr*	Buffer to store the Counter Id of the newly created counter (must not be null).
in	*CounterName*	The Name of the generic 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.
CFE_ES_ERR_DUPLICATE_NAME	Duplicate Name Error.
CFE_ES_NO_RESOURCE_IDS_AVAILABLE	Resource ID is not available.

See also

CFE_ES_IncrementGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGen↩Count, 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	*Counter↩Id*	The Counter to be set.
in	*Count*	The new value of the 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_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	*Filters*	Pointer to an array of event message filters, or NULL if no filtering is desired. The structure of an event message filter depends on the FilterScheme selected. (see Filter Schemes mentioned above)
in	*NumEventFilters*	The number of event message filters included in this call. This must be less than or equal to the maximum number of events allowed per application (CFE_PLATFORM_EVS_MAX_EVENT_FILTERS).
in	*FilterScheme*	The event filtering scheme that this application will use. For the first implementation of the event services, only filter type CFE_EVS_EventFilter_BINARY will be supported.

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 FM_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_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.

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	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.

Parameters

in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*Spec*	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

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 FM_AppInit(), FM_AppMain(), FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_Child↩DirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildInit(), FM_↩ChildLoop(), FM_ChildMoveCmd(), FM_ChildProcess(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), F↩M_ChildTask(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_IsValidCmdPktLength(), FM_NoopCmd(), F↩M_ProcessCmd(), FM_ProcessPkt(), FM_ResetCountersCmd(), FM_SetTableStateCmd(), FM_ValidateTable(), FM↩_VerifyChildTask(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_↩VerifyFileNoExist(), FM_VerifyFileNotOpen(), FM_VerifyNameValid(), and FM_VerifyOverwrite().

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	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.
in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*AppID*	The Application ID from which the event message should appear.
in	*Spec*	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 `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

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	*Time*	The time to include in the event. This will usually be a time returned by the function CFE_TIME_GetTime.
in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.
in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*Spec*	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_EVS_APP_NOT_REGISTERED	Application Not Registered.
CFE_EVS_APP_ILLEGAL_APP_ID	Illegal Application ID.
CFE_EVS_INVALID_PARAMETER	Invalid Pointer.

See also

CFE_EVS_SendEvent, CFE_EVS_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	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` 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	*Hdr*	Pointer to a variable of type CFE_FS_Header_t that will be cleared and initialized
in	*Description*	Initializes Header's Description (must not be null)
in	*SubType*	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	*Hdr*	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	*FileDes*	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	*FileDes*	File Descriptor obtained from a previous call to OS_OpenCreate that is associated with the file whose header is to be read.
in	*NewTimestamp*	A CFE_TIME_SysTime_t data structure containing the desired time to be put into the file's Standard cFE File Header.

Returns

Execution status, see cFE Return Code Defines, or OSAL status

Return values

CFE_STATUS_EXTERNAL_RESOURCE_FAIL	(return value only verified in coverage test) External failure.
CFE_SUCCESS	Successful execution.

Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in cFE Return Code Defines.

10.25.2.4 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	*FileDes*	File Descriptor obtained from a previous call to OS_OpenCreate that is associated with the file whose header is to be read.
out	*Hdr*	Pointer to a variable of type CFE_FS_Header_t (must not be null) that will be filled with the contents of the Standard cFE File Header. ∗Hdr is the contents of the Standard cFE File Header for the specified file.

Returns

Bytes read or error status from OSAL

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.

Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in cFE Return Code Defines.

See also

CFE_FS_ReadHeader

Referenced by FM_ChildDirListFileInit().

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

in,out	*Meta*	The background file write persistent state object (must not be null)

Returns

boolean value indicating if request is already pending

Return values

true	if request is pending
false	if request is not pending

10.26.2.2 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

`in,out`	*Meta*	The background file write persistent state object (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_FS_BAD_ARGUMENT	Bad Argument.
CFE_FS_INVALID_PATH	Invalid Path.
CFE_STATUS_REQUEST_ALREADY_PENDING	Request already pending.
CFE_SUCCESS	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	*OriginalPath*	The original path (must not be null)
out	*FileNameOnly*	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

NULL	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

NULL	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

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

Returns

Execution 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	*OutputBuffer*	Buffer to store result (must not be null).
in	*InputBuffer*	A input buffer that may contain a file name (e.g. from command) (must not be null).
in	*OutputBufSize*	Maximum Size of output buffer (must not be zero).
in	*InputBufSize*	Maximum Size of input buffer.
in	*DefaultInput*	Default value to use for input if InputBffer is empty
in	*DefaultPath*	Default value to use for pathname if omitted from input
in	*DefaultExtension*	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*MsgId*	MsgId that corresponds to message
in	*Size*	Total size of the message (used to set length field)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

Referenced by FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), and FM_ReportHK().

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_GetApId (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t *ApId)

  Gets the message application ID.
- CFE_Status_t CFE_MSG_SetApId (CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t ApId)

  Sets the message application ID.
- CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)

  Gets the message segmentation flag.
- CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)

  Sets the message segmentation flag.
- CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)

  Gets the message sequence count.
- CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)

  Sets the message sequence count.
- CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)

  Gets the next sequence count value (rolls over if appropriate)

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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*ApId*	Application ID (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.2 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*HasSecondary*	Has secondary header flag (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.3 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

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

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.4 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	*SeqCnt*	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*SegFlag*	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*SeqCnt*	Sequence count (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.

10.28.2.7 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Size*	Total message size (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.

Referenced by FM_IsValidCmdPktLength().

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

in	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Type*	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.

10.28.2.9 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*ApId*	Application ID

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*HasSecondary*	Has secondary header flag

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.28.2.11 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	*MsgPtr*	A pointer to the buffer that contains the message.
in	*Version*	Header version

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*SegFlag*	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

in,out	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*SeqCnt*	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

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

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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

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

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Version*	EDS Version (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.2 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Endian*	Endian (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.3 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*PlayFlag*	Playback Flag (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.4 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Subsystem*	Subsystem (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.5 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

in	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*System*	System (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.29.2.6 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*Version*	EDS Version

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*Endian*	Endian

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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

in,out	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*PlayFlag*	Playback Flag

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	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

in,out	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*Subsystem*	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*System*	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 ∗Fcn↩ Code)

  Gets the function code field from a message.
- CFE_Status_t CFE_MSG_GetMsgTime (const CFE_MSG_Message_t ∗MsgPtr, CFE_TIME_SysTime_t ∗Time)

  Gets the time field from a message.
- CFE_Status_t CFE_MSG_SetMsgTime (CFE_MSG_Message_t ∗MsgPtr, CFE_TIME_SysTime_t NewTime)

  Sets the time field in a message.

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

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.
CFE_MSG_WRONG_MSG_TYPE	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

in	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*FcnCode*	The function code from the message (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

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

Referenced by FM_DeleteFileCmd(), and FM_ProcessCmd().

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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*Time*	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

in,out	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*FcnCode*	The function code to include in the message.

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.

10.30.2.5 CFE_MSG_SetMsgTime()

```
CFE_Status_t CFE_MSG_SetMsgTime (
            CFE_MSG_Message_t * MsgPtr,
            CFE_TIME_SysTime_t NewTime )
```

Sets the time field in a message.

Description

This routine sets the time of a message. Most applications will want to use CFE_SB_TimeStampMsg instead of this function. But, when needed, this API can be used to set multiple messages with identical time stamps.

Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a time field, then this routine will do nothing to the message contents and will return CFE_MSG_WRONG_MSG_TYPE.
- Note default implementation of command messages do not have a time field.

Parameters

in,out	*MsgPtr*	A pointer to the message (must not be null).
in	*NewTime*	The time to include in the message. This will usually be a time from CFE_TIME_GetTime.

Returns

Execution status, see cFE Return Code Defines

Return values

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

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	*MsgPtr*	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	*IsValid*	Checksum validation result (must not be null) • true - valid • false - invalid or not supported/implemented

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.

10.31 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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
out	*MsgId*	Message id (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

Referenced by FM_ProcessPkt().

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	*Msg↩ Id*	Message id
out	*Type*	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	*MsgPtr*	A pointer to the buffer that contains the message (must not be null).
in	*MsgId*	Message id

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_MSG_BAD_ARGUMENT	Error - bad argument.

10.32 cFE Pipe Management APIs

Functions

- CFE_Status_t CFE_SB_CreatePipe (CFE_SB_PipeId_t ∗PipeIdPtr, uint16 Depth, const char ∗PipeName)

  Creates a new software bus pipe.
- CFE_Status_t CFE_SB_DeletePipe (CFE_SB_PipeId_t PipeId)

  Delete a software bus pipe.
- CFE_Status_t CFE_SB_PipeId_ToIndex (CFE_SB_PipeId_t PipeID, uint32 ∗Idx)

  Obtain an index value correlating to an SB Pipe ID.
- CFE_Status_t CFE_SB_SetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 Opts)

  Set options on a pipe.
- CFE_Status_t CFE_SB_GetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 ∗OptsPtr)

  Get options on a pipe.
- CFE_Status_t CFE_SB_GetPipeName (char ∗PipeNameBuf, size_t PipeNameSize, CFE_SB_PipeId_t PipeId)

  Get the pipe name for a given id.
- CFE_Status_t CFE_SB_GetPipeIdByName (CFE_SB_PipeId_t ∗PipeIdPtr, 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_PipeId_t * PipeIdPtr,
            uint16 Depth,
            const char * PipeName )
```

Creates a new software bus pipe.

Description

This routine creates and initializes an input pipe that the calling application can use to receive software bus messages. By default, no messages are routed to the new pipe. So, the application must use CFE_SB_Subscribe to specify which messages it wants to receive on this pipe.

Assumptions, External Events, and Notes:

None

Parameters

out	*PipeIdPtr*	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.
in	*Depth*	The maximum number of messages that will be allowed on this pipe at one time.
in	*PipeName*	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 FM_AppInit().

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	*PipeId*	The pipe ID (obtained previously from CFE_SB_CreatePipe) of the pipe to be deleted.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

CFE_SB_CreatePipe CFE_SB_GetPipeOpts CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

10.32.2.3 CFE_SB_GetPipeIdByName()

```
CFE_Status_t CFE_SB_GetPipeIdByName (
            CFE_SB_PipeId_t * PipeIdPtr,
            const char * PipeName )
```

Get pipe id by pipe name.

Description

This routine finds the pipe id for a pipe name.

Parameters

in	*PipeName*	The name of the pipe (must not be null).
out	*PipeIdPtr*	The PipeId for that name (must not be null).

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_PIPEOPTS_IGNOREMINE

10.32.2.4 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	*PipeNameBuf*	The buffer to receive the pipe name (must not be null).
in	*PipeNameSize*	The size (in chars) of the PipeName buffer (must not be zero).
in	*PipeId*	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	*PipeId*	The pipe ID of the pipe to get options from.
out	*OptsPtr*	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	*PipeID*	Pipe ID to convert
out	*Idx*	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	*PipeId*	The pipe ID of the pipe to set options on.
in	*Opts*	A bit field of options: cFE SB Pipe options

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_GetPipeOpts CFE_SB_GetPipeIdByName CFE_SB_PIPEOPTS_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	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.

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 FM_AppInit().

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	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*Quality*	The requested Quality of Service (QoS) required of the messages. Most callers will use CFE_SB_DEFAULT_QOS for this parameter.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_MAX_MSGS_MET	(return value only verified in coverage test) Max Messages Met.
CFE_SB_MAX_DESTS_MET	Max Destinations Met.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_BUF_ALOC_ERR	(return value only verified in coverage test) Buffer Allocation Error.

See also

CFE_SB_Subscribe, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

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	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_MAX_MSGS_MET	(return value only verified in coverage test) Max Messages Met.
CFE_SB_MAX_DESTS_MET	Max Destinations Met.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_BUF_ALOC_ERR	(return value only verified in coverage test) Buffer Allocation Error.

See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

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	*Msg↩ Id*	The message ID of the message to be unsubscribed.
in	*Pipe↩ Id*	The pipe ID of the pipe the subscribed message should no longer be sent to.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.

See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_UnsubscribeLocal

10.33.2.5 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	*MsgId*	The message ID of the message to be unsubscribed.
in	*PipeId*	The pipe ID of the pipe the subscribed message should no longer be sent to.

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	*BufPtr*	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	*PipeId*	The pipe ID of the pipe containing the message to be obtained.
in	*TimeOut*	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to CFE_SB_POLL for a non-blocking receive or CFE_SB_PEND_FOREVER to wait forever for a message to arrive.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_SB_BAD_ARGUMENT	Bad Argument.
CFE_SB_TIME_OUT	Time Out.
CFE_SB_PIPE_RD_ERR	(return value only verified in coverage test) Pipe Read Error.
CFE_SB_NO_MESSAGE	No Message.

Referenced by FM_AppMain().

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	*MsgPtr*	A pointer to the message to be sent (must not be null). This must point to the first byte of the message header.
in	*IncrementSequenceCount*	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.
CFE_SB_BUF_ALOC_ERR	(return value only verified in coverage test) Buffer Allocation Error.

Referenced by FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), and FM_ReportHK().

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	*BufPtr*	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	*BufPtr*	A pointer to the buffer to be sent (must not be null).
in	*IncrementSequenceCount*	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	*MsgPtr*	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`	*DestStringPtr*	Pointer to destination buffer (must not be null)
`in`	*SourceStringPtr*	Pointer to source buffer (component of SB message definition)
`in`	*DefaultString*	Default string to use if source is empty
`in`	*DestMaxSize*	Size of destination storage buffer (must not be zero)
`in`	*SourceMaxSize*	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`	*DestStringPtr*	Pointer to destination buffer (component of SB message definition) (must not be null)
`in`	*SourceStringPtr*	Pointer to source buffer (must not be null)
`in`	*DestMaxSize*	Size of destination buffer as defined by the message definition
`in`	*SourceMaxSize*	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	*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.
in	*DataLength*	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	*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.

Referenced by FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), and FM_ReportHK().

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

true	Message ID is within the valid range
false	Message ID is not within the valid range

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

true	Message IDs are Equal
false	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 FM_ProcessPkt().

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 FM_AppInit(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_Get↩OpenFilesCmd(), and FM_ReportHK().

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	*TblHandlePtr*	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 address specified by TblHandlePtr.
in	*Name*	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	*Size*	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	*TblOptionFlags*	Flag bits indicating selected options for table. A bitwise OR of the following option flags: • CFE_TBL_OPT_DEFAULT - The default setting for table options is a combination of CFE_TBL_OPT_SNGL_BUFFER and CFE_TBL_OPT_LOAD_DUMP. See below for a description of these two options. This option is mutually exclusive with the CFE_TBL_OPT_DBL_BUFFER, CFE_TBL_OPT_DUMP_ONLY and CFE_TBL_OPT_USR_DEF_ADDR options. • CFE_TBL_OPT_SNGL_BUFFER - When this option is selected, the table will use a shared session table for performing table modifications and a memory copy from the session table to the "active" table buffer will occur when the table is updated. This is the preferred option since it will minimize memory usage. This option is mutually exclusive with the CFE_TBL_OPT_DBL_BUFFER option • CFE_TBL_OPT_DBL_BUFFER - When this option is selected, two instances of the table are created. One is considered the "active" table and the other the "inactive" table. Whenever table modifications occur, they do not require the use of a common session table. Modifications occur in the "inactive" buffer. Then, when it is time to update the table, the pointer to the "active" table is changed to point to the "inactive" buffer thus making it the new "active" buffer. This feature is most useful for time critical applications (ie - interrupt service routines, etc). This option is mutually exclusive with the CFE_TBL_OPT_SNGL_BUFFER and CFE_TBL_OPT_DEFAULT option. • CFE_TBL_OPT_LOAD_DUMP - When this option is selected, the Table Service is allowed to perform all operations on the specified table. This option is mutually exclusive with the CFE_TBL_OPT_DUMP_ONLY option. • CFE_TBL_OPT_DUMP_ONLY - When this option is selected, the Table Service will not perform table loads to this table. This does not prevent, however, a task from writing to the table via an address obtained with the CFE_TBL_GetAddress API function. This option is mutually exclusive with the CFE_TBL_OPT_LOAD_DUMP and CFE_TBL_OPT_DEFAULT options. If the Application wishes to specify their own block of memory as the Dump Only table, they need to also include the CFE_TBL_OPT_USR_DEF_ADDR option explained below. • CFE_TBL_OPT_NOT_USR_DEF - When this option is selected, Table Services allocates memory for the table and, in the case of a double buffered table, it allocates the same amount of memory again for the second buffer. This option is mutually exclusive with the CFE_TBL_OPT_USR_DEF_ADDR option. • CFE_TBL_OPT_USR_DEF_ADDR- When this option is selected, the Table Service will not allocate memory for the table. Table Services will require the Application to identify the location of the active table buffer via the CFE_TBL_Load function. This option implies the CFE_TBL_OPT_DUMP_ONLY and the CFE_TBL_OPT_SNGL_BUFFER options and is mutually exclusive of the CFE_TBL_OPT_DBL_BUFFER option. • CFE_TBL_OPT_CRITICAL- When this option is selected, the Table Service will automatically allocate space in the Critical Data Store (CDS) for the table and 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

Parameters

in	*TblValidationFuncPtr*	is a pointer to a function that will be executed in the context of the Table Management Service when the contents of a table need to be validated. If set to NULL, then the Table Management Service will assume any data is valid. If the value is not NULL, it must be a pointer to a function with the following prototype: **int32 CallbackFunc(void ∗TblPtr);** where **TblPtr** will be a pointer to the table data that is to be verified. When the function returns CFE_SUCCESS, the data is considered valid and ready for a commit. When the function returns a negative value, the data is considered invalid and an Event Message will be issued containing the returned value. If the function should return a positive number, the table is considered invalid and the return code is considered invalid. Validation functions **must** return either CFE_SUCCESS or a negative number (whose value is at the developer's discretion). The validation function will be executed in the Application's context so that Event Messages describing the validation failure are possible from within the function.

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 FM_TableInit().

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	*TblHandlePtr*	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	*TblName*	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	*TblHandle*	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	*TblHandle*	Handle of Table to be dumped.

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_INFO_DUMP_PENDING	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	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be loaded.
in	*SrcType*	Flag indicating the nature of the given `SrcDataPtr` below. This value can be any one of the following: • CFE_TBL_SRC_FILE - File source When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table. • CFE_TBL_SRC_ADDRESS - Address source When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the CFE_TBL_Register function Size parameter.

Parameters

in	*SrcDataPtr*	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 FM_TableInit().

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	*TblHandle*	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 FM_AcquireTablePointers().

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	*TblHandle*	Handle of Table that was modified.

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_Manage

Referenced by FM_SetTableStateCmd().

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	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be updated.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_NO_UPDATE_PENDING	No Update Pending.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

CFE_TBL_Load, CFE_TBL_Validate, CFE_TBL_Manage

10.40.2.6 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	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be managed.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_NO_VALIDATION_PENDING	

Return values

CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

CFE_TBL_Update, CFE_TBL_Manage, CFE_TBL_Load

10.41 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 Tbl↩Handles[ ])

  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_Get↩Addresses.

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 CF↩E_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	*TblPtr*	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	*TblHandle*	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 FM_AcquireTablePointers().

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_Get↩Address.

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	*TblPtrs*	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.
in	*NumTables*	Size of TblPtrs and TblHandles arrays.
in	*TblHandles*	Array of Table Handles, previously obtained from CFE_TBL_Register or CFE_TBL_Share, of those tables whose start addresses are to be obtained.

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	*TblHandle*	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 FM_ReleaseTablePointers().

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	*NumTables*	Size of TblHandles array.
in	*TblHandles*	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	*TblInfoPtr*	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	*TblName*	The application specific name (must not be null) of the table of the form "AppName.RawTableName", where RawTableName is the name specified in the CFE_TBL_Register API call. Example: "ACS.TamParams" for a table called "TamParams" that was registered by the application called "ACS".

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_ERR_INVALID_NAME	Invalid Name.
CFE_TBL_BAD_ARGUMENT	Bad Argument.

See also

CFE_TBL_GetStatus

10.42.2.2 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	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be managed.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TBL_INFO_UPDATE_PENDING	Update Pending.

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	*TblHandle*	Handle of Table with which the message should be associated.
in	*MsgId*	Message ID to be used in notification message sent by Table Services.
in	*CommandCode*	Command Code value to be placed in secondary header of message sent by Table Services.
in	*Parameter*	Application defined value to be passed as a parameter in the message sent by Table
		Services. Suggested use includes an application's table index that allows the same MsgId and Command Code to be used for all table management notifications.

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_ES_ERR_RESOURCEID_NOT_VALID	Resource ID is not valid.
CFE_TBL_ERR_NO_ACCESS	No Access.
CFE_TBL_ERR_INVALID_HANDLE	Invalid Handle.

See also

CFE_TBL_Register

10.43 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.

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.

Referenced by FM_TableInit().

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.

Referenced by FM_TableInit().

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_GetME↩Tsubsecs, 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_GetME↩Tseconds, 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_xxxxxx) == CFE_TIME_FLAG_xxxxxx)` then the following definition of the `CFE_TIME_FLAG_xxxxxx` 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	*Time1*	The first time to be added.
in	*Time2*	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_Sys↩Time_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	*SubSeconds*	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 spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is CFE_PLATFORM_TIME_CFG_SERVER which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is CFE_PLATFORM_TIME_CFG_SOURCE which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is CFE_PLATFORM_TIME_CFG_SRC_GPS, which indicates that the external time data consists of a time value relative to a known epoch, plus a leap seconds value.

Parameters

in	*NewTime*	The MET value at the next (or previous) 1 Hz tone signal.
in	*NewLeaps*	The Leap Seconds value used to calculate time as UTC.

See also

CFE_TIME_ExternalTone, CFE_TIME_ExternalMET, CFE_TIME_ExternalTime

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	*NewMET*	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 spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is CFE_PLATFORM_TIME_CFG_SERVER which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is CFE_PLATFORM_TIME_CFG_SOURCE which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is CFE_PLATFORM_TIME_CFG_SRC_TIME, which indicates that the external time data consists of a time value relative to a known epoch.

Parameters

in	*NewTime*	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	*CallbackFuncPtr*	Function to call at synchronization interval (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS	Too Many Sync Callbacks.
CFE_TIME_BAD_ARGUMENT	Bad Argument.

See also

CFE_TIME_UnregisterSynchCallback

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	*CallbackFuncPtr*	Function to remove from synchronization call list (must not be null)

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.
CFE_TIME_CALLBACK_NOT_REGISTERED	Callback Not Registered.
CFE_TIME_BAD_ARGUMENT	Bad Argument.

See also

CFE_TIME_RegisterSynchCallback

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`	*PrintBuffer*	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`	*TimeToPrint*	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_OFF↩SET = 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_R↩ESOURCE_BASE_OFFSET = OS_OBJECT_TYPE_USER + 6, CFE_RESOURCEID_CONFIGID_BASE_OF↩FSET = OS_OBJECT_TYPE_USER + 7 }
- enum {
  CFE_ES_TASKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_O↩FFSET), CFE_ES_APPID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BA↩SE_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_E↩S_COUNTID_BASE_OFFSET),
  CFE_ES_POOLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_O↩FFSET), CFE_ES_CDSBLOCKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CD↩SBLOCKID_BASE_OFFSET), CFE_SB_PIPEID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOUR↩CEID_SB_PIPEID_RESOURCE_BASE_OFFSET), CFE_CONFIGID_BASE = CFE_RESOURCEID_MAKE_B↩ASE(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

out	*sem_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*sem_name*	the name of the new resource to create (must not be null)
in	*sem_initial_value*	the initial value of the binary semaphore
in	*options*	Reserved for future use, should be passed as 0.

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if sen name or sem_id are NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NO_FREE_IDS	if all of the semaphore ids are taken
OS_ERR_NAME_TAKEN	if this is already the name of a binary semaphore
OS_SEM_FAILURE	if the OS call failed (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	*sem↩ _id*	The object ID to delete

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid binary semaphore
OS_SEM_FAILURE	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	*sem↩ _id*	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	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	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	*sem_id*	The object ID to operate on
out	*bin_prop*	The property object buffer to fill (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid semaphore
OS_INVALID_POINTER	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	*sem↩ _id*	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)

10.53.2.7 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	*sem↩ _id*	The object ID to operate on

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	the Id passed in is not a valid binary semaphore
OS_SEM_FAILURE	if an unspecified failure occurs (return value only verified in coverage test)

10.53.2.8 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	*sem↩ _id*	The object ID to operate on
in	*msecs*	The maximum amount of time to block, in milliseconds

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_SEM_TIMEOUT	if semaphore was not relinquished in time
OS_ERR_INVALID_ID	if the ID passed in is not a valid semaphore ID
OS_SEM_FAILURE	if an unspecified failure occurs (return value only verified in coverage test)

10.54 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	*time_struct*	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	*time_struct*	An OS_time_t containing the current time (must not be null)

Returns

Set local time status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if time_struct is null

10.55.2.3 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	*time1*	The first interval
in	*time2*	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	*seconds*	Whole number of seconds
in	*microseconds*	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	*seconds*	Whole number of seconds
in	*milliseconds*	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	*seconds*	Whole number of seconds
in	*subseconds*	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	*tm*	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_usec" field of POSIX-style "struct timeval" values.

See also

OS_TimeGetTotalSeconds()

Parameters

in	*tm*	Time interval value

Returns

Number of microseconds in time interval

Definition at line 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	*tm*	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 bsps, unit tests, psps, 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

OS_SUCCESS	Successful execution.
OS_ERROR	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	*flag*	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	*sem_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*sem_name*	the name of the new resource to create (must not be null)
in	*sem_initial_value*	the initial value of the counting semaphore
in	*options*	Reserved for future use, should be passed as 0.

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if sen name or sem_id are NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NO_FREE_IDS	if all of the semaphore ids are taken
OS_ERR_NAME_TAKEN	if this is already the name of a counting semaphore
OS_INVALID_SEM_VALUE	if the semaphore value is too high (return value only verified in coverage test)
OS_SEM_FAILURE	if an unspecified implementation error occurs (return value only verified in coverage test)

Referenced by FM_ChildInit().

10.57.2.2 OS_CountSemDelete()

```
int32 OS_CountSemDelete (
            osal_id_t sem_id )
```

Deletes the specified counting Semaphore.

Parameters

in	*sem↩ _id*	The object ID to delete

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid counting semaphore
OS_SEM_FAILURE	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	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	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

10.57.2.4 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	*sem_id*	The object ID to operate on
out	*count_prop*	The property object buffer to fill (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid semaphore
OS_INVALID_POINTER	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

in	*sem↩ _id*	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 counting semaphore
OS_SEM_FAILURE	if an unspecified implementation error occurs (return value only verified in coverage test)

Referenced by FM_InvokeChildTask().

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	*sem↩ _id*	The object ID to operate on

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	the Id passed in is not a valid counting semaphore
OS_SEM_FAILURE	if an unspecified implementation error occurs (return value only verified in coverage test)

Referenced by FM_ChildLoop().

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	*sem↩ _id*	The object ID to operate on
in	*msecs*	The maximum amount of time to block, in milliseconds

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_SEM_TIMEOUT	if semaphore was not relinquished in time
OS_ERR_INVALID_ID	if the ID passed in is not a valid semaphore ID
OS_SEM_FAILURE	if an unspecified 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	*dir_id*	The handle ID of the directory

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the directory handle is invalid

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), and FM_ChildDirList↩PktCmd().

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	*dir↩_id*	Location to store handle ID of the directory (must not be null)
in	*path*	The directory to open (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if dir_id or path is NULL
OS_FS_ERR_PATH_TOO_LONG	if the path argument exceeds the maximum length
OS_FS_ERR_PATH_INVALID	if the path argument is not valid
OS_ERROR	if the directory could not be opened

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), and FM_ChildDirList↩PktCmd().

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	*dir↩ _id*	The handle ID of the directory
out	*dirent*	Buffer to store directory entry information (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if dirent argument is NULL
OS_ERR_INVALID_ID	if the directory handle is invalid
OS_ERROR	at the end of the directory or if the OS call otherwise fails

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileLoop(), and FM_ChildDirList↩ PktCmd().

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	*dir↩ _id*	The handle ID of the directory

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the directory handle is invalid

Referenced by FM_ChildDeleteAllCmd().

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	*path*	The new directory name (must not be null)
in	*access*	The permissions for the directory (reserved for future use)

Note

Current implementations do not utilize the "access" parameter. Applications should still pass the intended value (OS_READ_WRITE or OS_READ_ONLY) to be compatible with future implementations.

Returns

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)

Referenced by FM_ChildCreateDirCmd().

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	*path*	The directory to remove

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if path is NULL
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_PATH_TOO_LONG	
OS_ERROR	if the directory remove operation failed (return value only verified in coverage test)

Referenced by FM_ChildDeleteDirCmd().

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)

File error.

- #define OS_ERR_NOT_IMPLEMENTED (-28)

  Not implemented.
- #define OS_TIMER_ERR_INVALID_ARGS (-29)

  Timer invalid arguments.
- #define OS_TIMER_ERR_TIMER_ID (-30)

  Timer ID error.
- #define OS_TIMER_ERR_UNAVAILABLE (-31)

  Timer unavailable.
- #define OS_TIMER_ERR_INTERNAL (-32)

  Timer internal error.
- #define OS_ERR_OBJECT_IN_USE (-33)

  Object in use.
- #define OS_ERR_BAD_ADDRESS (-34)

  Bad address.
- #define OS_ERR_INCORRECT_OBJ_STATE (-35)

  Incorrect object state.
- #define OS_ERR_INCORRECT_OBJ_TYPE (-36)

  Incorrect object type.
- #define OS_ERR_STREAM_DISCONNECTED (-37)

  Stream disconnected.
- #define OS_ERR_OPERATION_NOT_SUPPORTED (-38)

  Requested operation not support on supplied object(s)
- #define OS_ERR_INVALID_SIZE (-40)

  Invalid Size.
- #define OS_ERR_OUTPUT_TOO_LARGE (-41)

  Size of output exceeds limit.
- #define OS_ERR_INVALID_ARGUMENT (-42)

  Invalid argument value (other than ID or size)
- #define OS_FS_ERR_PATH_TOO_LONG (-103)

  FS path too long.
- #define OS_FS_ERR_NAME_TOO_LONG (-104)

  FS name too long.
- #define OS_FS_ERR_DRIVE_NOT_CREATED (-106)

  FS drive not created.
- #define OS_FS_ERR_DEVICE_NOT_FREE (-107)

  FS device not free.
- #define OS_FS_ERR_PATH_INVALID (-108)

  FS path invalid.

10.59.1 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.

Referenced by FM_ChildLoop().

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.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_ChildSizeTimeMode(), FM_GetFilenameState(), FM_GetFreeSpaceCmd(), LoadOpenFileData(), and SearchOpenFileData().

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	*error_num*	Error number to convert
out	*err_name*	Buffer to store error string

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	if successfully converted to a string
OS_INVALID_POINTER	if err_name is NULL
OS_ERROR	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	*Status*	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.

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

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.

Referenced by FM_ChildConcatCmd(), and FM_ChildDirListFileInit().

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.

Referenced by FM_ChildConcatCmd().

10.62.2.3 OS_SEEK_SET

```
#define OS_SEEK_SET 0
```

Seek offset set

Definition at line 43 of file osapi-file.h.

Referenced by FM_ChildDirListFileLoop().

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	*path*	File to change (must not be null)
in	*access_mode*	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

Referenced by FM_ChildSetPermissionsCmd().

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	*filedes*	The handle ID to operate on

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
OS_ERROR	if an unexpected/unhandled error occurs (return value only verified in coverage test)

Referenced by FM_ChildConcatCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListFileInit(), and FM_ChildFileInfo↩Cmd().

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	*Filename*	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	*src*	The source file to operate on (must not be null)
in	*dest*	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 accessed
OS_INVALID_POINTER	if src or dest are NULL
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_PATH_TOO_LONG	if the paths given are too long to be stored locally
OS_FS_ERR_NAME_TOO_LONG	if the dest name is too long to be stored locally

Referenced by FM_ChildConcatCmd(), and FM_ChildCopyCmd().

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	*filedes*	The handle ID to operate on
out	*fd_prop*	Storage buffer for file information (must not be null)

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_INVALID_POINTER	if the fd_prop argument is NULL

Referenced by LoadOpenFileData(), and SearchOpenFileData().

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	*Filename*	The file to operate on (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	if the file is open
OS_ERROR	if the file is not open
OS_INVALID_POINTER	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	*filedes*	The handle ID to operate on
in	*offset*	The file offset to seek to
in	*whence*	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

OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
OS_ERROR	if OS call failed (return value only verified in coverage test)

Referenced by FM_ChildConcatCmd(), and FM_ChildDirListFileLoop().

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	*src*	The source file to operate on (must not be null)
in	*dest*	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

Referenced by FM_ChildMoveCmd().

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	*filedes*	The handle ID (OS_OBJECT_ID_UNDEFINED on failure) (must not be null)
in	*path*	File name to create or open (must not be null)
in	*flags*	The file permissions - see OS_file_flag_t
in	*access_mode*	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

Referenced by FM_ChildConcatCmd(), FM_ChildDirListFileInit(), and FM_ChildFileInfoCmd().

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	*filedes*	The handle ID to operate on
out	*buffer*	Storage location for file data (must not be null)
in	*nbytes*	Maximum number of bytes to read (must not be zero)

Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

Returns

A non-negative byte count or appropriate error code, see OSAL Return Code Defines

Return values

OS_INVALID_POINTER	if buffer is a null pointer
OS_ERR_INVALID_SIZE	if the passed-in size is not valid
OS_ERROR	if OS call failed (return value only verified in coverage test)
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
0	if at end of file/stream data

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

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	*path*	The file to operate on (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	if there is no device or the driver returns error
OS_INVALID_POINTER	if path is NULL
OS_FS_ERR_PATH_TOO_LONG	if path is too long to be stored locally
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_NAME_TOO_LONG	if the name of the file to remove is too long

Referenced by FM_ChildConcatCmd(), FM_ChildDeleteAllCmd(), and FM_ChildDeleteCmd().

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	*old_filename*	The original filename (must not be null)
in	*new_filename*	The desired filename (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 opened or renamed.
OS_INVALID_POINTER	if old or new are NULL
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_FS_ERR_PATH_TOO_LONG	if the paths given are too long to be stored locally
OS_FS_ERR_NAME_TOO_LONG	if the new name is too long to be stored locally

Referenced by FM_ChildRenameCmd().

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

in	*path*	The file to operate on (must not be null)
out	*filestats*	Buffer to store file information (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if path or filestats is NULL
OS_FS_ERR_PATH_TOO_LONG	if the path is too long to be stored locally
OS_FS_ERR_NAME_TOO_LONG	if the name of the file is too long to be stored
OS_FS_ERR_PATH_INVALID	if path cannot be parsed
OS_ERROR	if the OS call failed

Referenced by FM_ChildSizeTimeMode(), and FM_GetFilenameState().

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	*filedes*	The handle ID to operate on
out	*buffer*	Storage location for file data (must not be null)
in	*nbytes*	Maximum number of bytes to read (must not be zero)
in	*timeout*	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	*filedes*	The handle ID to operate on
in	*buffer*	Source location for file data (must not be null)
in	*nbytes*	Maximum number of bytes to read (must not be zero)
in	*timeout*	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	*filedes*	The handle ID to operate on
in	*buffer*	Source location for file data (must not be null)
in	*nbytes*	Maximum number of bytes to read (must not be zero)

Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

Returns

A non-negative byte count or appropriate error code, see OSAL Return Code Defines

Return values

OS_INVALID_POINTER	if buffer is NULL
OS_ERR_INVALID_SIZE	if the passed-in size is not valid
OS_ERROR	if OS call failed (return value only verified in coverage test)
OS_ERR_INVALID_ID	if the file descriptor passed in is invalid
0	if file/stream cannot accept any more data

Referenced by FM_ChildConcatCmd(), FM_ChildDirListFileInit(), and FM_ChildDirListFileLoop().

10.64 OSAL File System Level APIs

Functions

- int32 OS_FileSysAddFixedMap (osal_id_t ∗filesys_id, const char ∗phys_path, const char ∗virt_path)

  Create a fixed mapping between an existing directory and a virtual OSAL mount point.
- int32 OS_mkfs (char ∗address, const char ∗devname, const char ∗volname, size_t blocksize, osal_blockcount_t numblocks)

  Makes a file system on the target.
- int32 OS_mount (const char ∗devname, const char ∗mountpoint)

  Mounts a file system.
- int32 OS_initfs (char ∗address, const char ∗devname, const char ∗volname, size_t blocksize, osal_blockcount_t numblocks)

  Initializes an existing file system.
- int32 OS_rmfs (const char ∗devname)

  Removes a file system.
- int32 OS_unmount (const char ∗mountpoint)

  Unmounts a mounted file system.
- int32 OS_FileSysStatVolume (const char ∗name, OS_statvfs_t ∗statbuf)

  Obtains information about size and free space in a volume.
- int32 OS_chkfs (const char ∗name, bool repair)

  Checks the health of a file system and repairs it if necessary.
- int32 OS_FS_GetPhysDriveName (char ∗PhysDriveName, const char ∗MountPoint)

  Obtains the physical drive name associated with a mount point.
- int32 OS_TranslatePath (const char ∗VirtualPath, char ∗LocalPath)

  Translates an OSAL Virtual file system path to a host Local path.
- int32 OS_GetFsInfo (os_fsinfo_t ∗filesys_info)

  Returns information about the file system.

10.64.1 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	*name*	The device/path to operate on (must not be null)
in	*repair*	Whether to also repair inconsistencies

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution. (return value only verified in coverage test)
OS_INVALID_POINTER	Name is NULL
OS_ERR_NOT_IMPLEMENTED	Not implemented.
OS_FS_ERR_PATH_TOO_LONG	if the name is too long
OS_ERROR	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	*filesys_id*	A buffer to store the ID of the file system mapping (must not be null)
in	*phys_path*	The native system directory (an existing mount point) (must not be null)
in	*virt_path*	The virtual mount point of this filesystem (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_FS_ERR_PATH_TOO_LONG	if the overall phys_path is too long
OS_ERR_NAME_TOO_LONG	if the phys_path basename (filesystem name) is too long
OS_INVALID_POINTER	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

in	*name*	The device/path to operate on (must not be null)
out	*statbuf*	Output structure to populate (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if name or statbuf is NULL
OS_FS_ERR_PATH_TOO_LONG	if the name is too long
OS_ERROR	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

Referenced by FM_GetFreeSpaceCmd().

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	*PhysDriveName*	Buffer to store physical drive name (must not be null)
in	*MountPoint*	OSAL mount point (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if either parameter is NULL
OS_ERR_NAME_NOT_FOUND	if the MountPoint is not mounted in OSAL
OS_FS_ERR_PATH_TOO_LONG	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	*filesys_info*	Buffer to store filesystem information (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if filesys_info 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	*address*	The address at which to start the new disk. If address == 0, then space will be allocated by the OS
in	*devname*	The underlying kernel device to use, if applicable. (must not be null)
in	*volname*	The name of the volume (see note) (must not be null)
in	*blocksize*	The size of a single block on the drive
in	*numblocks*	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 are NULL
OS_FS_ERR_PATH_TOO_LONG	if the name is too long
OS_FS_ERR_DEVICE_NOT_FREE	if the volume table is full
OS_FS_ERR_DRIVE_NOT_CREATED	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	*address*	The address at which to start the new disk. If address == 0 space will be allocated by the OS.
in	*devname*	The underlying kernel device to use, if applicable. (must not be null)
in	*volname*	The name of the volume (see note) (must not be null)
in	*blocksize*	The size of a single block on the drive
in	*numblocks*	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	*devname*	The name of the drive to mount. devname is the same from OS_mkfs (must not be null)
in	*mountpoint*	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	*devname*	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

OS_INVALID_POINTER	if devname is NULL
OS_FS_ERR_PATH_TOO_LONG	if the devname is too long
OS_ERR_NAME_NOT_FOUND	if the devname does not exist in OSAL
OS_ERROR	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

10.64.2.10 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	*VirtualPath*	OSAL virtual path name (must not be null)
out	*LocalPath*	Buffer to store native/translated path name (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if either parameter is NULL
OS_FS_ERR_NAME_TOO_LONG	if the filename component is too long
OS_FS_ERR_PATH_INVALID	if either parameter cannot be interpreted as a path
OS_FS_ERR_PATH_TOO_LONG	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	*mountpoint*	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

out	*heap_prop*	Storage buffer for heap info

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if the heap_prop 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.

Referenced by LoadOpenFileData(), and SearchOpenFileData().

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_ForEachObjectOfType (osal_objtype_t objtype, osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void ∗callback_arg)

  call the supplied callback function for valid object IDs of a specific type

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	*object_id*	The object ID to operate on
out	**ArrayIndex*	The Index to return (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

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().

10.67.2.2 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	*creator_id*	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	*callback_ptr*	Function to invoke for each matching object ID
in	*callback_arg*	Opaque Argument to pass to callback function (may be NULL)

Referenced by FM_GetFilenameState(), FM_GetOpenFilesData(), and OS_ObjectIdDefined().

10.67.2.3 OS_ForEachObjectOfType()

```
void OS_ForEachObjectOfType (
            osal_objtype_t objtype,
```

```
            osal_id_t creator_id,
            OS_ArgCallback_t callback_ptr,
            void * callback_arg )
```

call the supplied callback function for valid object IDs of a specific type

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	*objtype*	The type of objects to iterate
in	*creator_id*	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	*callback_ptr*	Function to invoke for each matching object ID
in	*callback_arg*	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	*object_id*	The object ID to operate on
out	*buffer*	Buffer in which to store the name (must not be null)
in	*buffer_size*	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	*object↩ _id*	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 LoadOpenFileData(), OS_ObjectIdDefined(), and SearchOpenFileData().

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	*object↩ _id*	The first object ID

Definition at line 147 of file osapi-idmap.h.

References OS_ConvertToArrayIndex(), OS_ForEachObject(), OS_ForEachObjectOfType(), OS_GetResourceName(), OS_IdentifyObject(), OS_ObjectIdToArrayIndex(), and OS_ObjectIdToInteger().

Referenced by FM_InvokeChildTask(), and FM_VerifyChildTask().

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	*object_id1*	The first object ID
in	*object_id2*	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	*value*	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	*idtype*	The object type to convert
in	*object_id*	The object ID to operate on
out	∗*ArrayIndex*	The Index to return (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

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	*object↩ _id*	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	*module_id*	OSAL ID of the previously the loaded module
out	*module_info*	Buffer to store module information (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the module id invalid

Return values

OS_INVALID_POINTER	if the pointer to the ModuleInfo structure is invalid
OS_ERROR	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	*module_id*	Non-zero OSAL ID corresponding to the loaded module
in	*module_name*	Name of module (must not be null)
in	*filename*	File containing the object code to load (must not be null)
in	*flags*	Options for the loaded module

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if one of the parameters is NULL
OS_ERR_NO_FREE_IDS	if the module table is full
OS_ERR_NAME_TAKEN	if the name is in use
OS_ERR_NAME_TOO_LONG	if the module_name is too long
OS_FS_ERR_PATH_INVALID	if the filename argument is not valid
OS_ERROR	if an other/unspecified error occurs (return value only verified in coverage test)

10.68.2.3 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	*module_id*	Module ID that should contain the symbol
out	*symbol_address*	Set to the address of the symbol (must not be null)
in	*symbol_name*	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	*module↩ _id*	OSAL ID of the previously the loaded module

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the module id invalid
OS_ERROR	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	*symbol_address*	Set to the address of the symbol (must not be null)
in	*symbol_name*	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

10.68.2.6 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	*filename*	File to write to (must not be null)
in	*size_limit*	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	*sem_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*sem_name*	the name of the new resource to create (must not be null)
in	*options*	reserved for future use. Should be passed as 0.

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if sem_id or sem_name are NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NO_FREE_IDS	if there are no more free mutex Ids
OS_ERR_NAME_TAKEN	if there is already a mutex with the same name
OS_SEM_FAILURE	if the OS call failed (return value only verified in coverage test)

Referenced by FM_ChildInit().

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	*sem↩ _id*	The object ID to delete

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid mutex
OS_SEM_FAILURE	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	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	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

10.69.2.4 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	*sem_id*	The object ID to operate on
out	*mut_prop*	The property object buffer to fill (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid semaphore
OS_INVALID_POINTER	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	*sem↩ _id*	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 valid mutex
OS_SEM_FAILURE	if an unspecified error occurs (return value only verified in coverage test)

Referenced by FM_ChildProcess(), and FM_InvokeChildTask().

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	*sem↩ _id*	The object ID to operate on

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	the id passed in is not a valid mutex
OS_SEM_FAILURE	if an unspecified error occurs (return value only verified in coverage test)

Referenced by FM_ChildProcess(), and FM_InvokeChildTask().

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	*host_name*	Buffer to hold name information (must not be null)
in	*name_len*	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 thats that will allow non-polled print statements for the real time systems.

Operates in a manner similar to the printf() call defined by the standard C library 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

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	*queue_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*queue_name*	the name of the new resource to create (must not be null)
in	*queue_depth*	the maximum depth of the queue
in	*data_size*	the size of each entry in the queue (must not be zero)
in	*flags*	options for the queue (reserved for future use, pass as 0)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if a pointer passed in is NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NO_FREE_IDS	if there are already the max queues created
OS_ERR_NAME_TAKEN	if the name is already being used on another queue
OS_ERR_INVALID_SIZE	if data_size is 0
OS_QUEUE_INVALID_SIZE	if the queue depth exceeds the limit
OS_ERROR	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	*queue↩ _id*	The object ID to delete

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in does not exist
OS_ERROR	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	*queue_id*	The object ID to operate on
out	*data*	The buffer to store the received message (must not be null)
in	*size*	The size of the data buffer (must not be zero)
out	*size_copied*	Set to the actual size of the message (must not be null)
in	*timeout*	The maximum amount of time to block, or OS_PEND to wait forever

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the given ID does not exist
OS_INVALID_POINTER	if a pointer passed in is NULL
OS_QUEUE_EMPTY	if the Queue has no messages on it to be received
OS_QUEUE_TIMEOUT	if the timeout was OS_PEND and the time expired
OS_QUEUE_INVALID_SIZE	if the size copied from the queue was not correct
OS_ERROR	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	*queue_id*	will be set to the ID of the existing resource
in	*queue_name*	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	if the name or id pointers are NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NAME_NOT_FOUND	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	*queue_id*	The object ID to operate on
out	*queue_prop*	The property object buffer to fill (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if queue_prop is NULL
OS_ERR_INVALID_ID	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	*queue↩ _id*	The object ID to operate on
in	*data*	The buffer containing the message to put (must not be null)
in	*size*	The size of the data buffer (must not be zero)
in	*flags*	Currently reserved/unused, should be passed as 0

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the queue id passed in is not a valid queue
OS_INVALID_POINTER	if the data pointer is NULL
OS_QUEUE_INVALID_SIZE	if the data message is too large for the queue
OS_QUEUE_FULL	if the queue cannot accept another message
OS_ERROR	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	*Set*	Pointer to OS_FdSet object to operate on (must not be null)
in	*objid*	The handle ID to add to 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

10.73.2.2 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	*Set*	Pointer to OS_FdSet object to operate on (must not be null)
in	*objid*	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	*Set*	Pointer to OS_FdSet object to operate on (must not be null)
in	*objid*	The handle ID to check for in the set

Returns

Boolean set status

Return values

true	FdSet structure contains ID
false	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

out	*Set*	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

in,out	*ReadSet*	Set of handles to check/wait to become readable
in,out	*WriteSet*	Set of handles to check/wait to become writable
in	*msecs*	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_WR↩ ITABLE) 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`	*objid*	The handle ID to select on
`in,out`	*StateFlags*	State flag(s) (readable or writable) (must not be null)
`in`	*msecs*	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 the handle is readable and/or writable, as requested
OS_ERROR_TIMEOUT	If the handle did not become readable or writable within the timeout
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_ID	if the objid is not a valid handle

10.74 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	*Cmd*	Command to pass to shell (must not be null)
in	*filedes*	File to send output to.

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 Cmd argument is NULL
OS_ERR_INVALID_ID	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

out	*Addr*	The address buffer to initialize (must not be null)
in	*string*	The string to initialize the address from (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
OS_ERROR	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

out	*PortNum*	Buffer to store the port number (must not be null)
in	*Addr*	The network address buffer (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
OS_ERR_BAD_ADDRESS	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	*Addr*	The address buffer to initialize (must not be null)
in	*Domain*	The address family

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if Addr argument is NULL
OS_ERR_NOT_IMPLEMENTED	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	*Addr*	The network address buffer (must not be null)
in	*PortNum*	The port number to set

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL
OS_ERR_BAD_ADDRESS	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	*buffer*	Buffer to hold the output string (must not be null)
in	*buflen*	Maximum length of the output string (must not be zero)
in	*Addr*	The network address buffer to convert (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
OS_ERR_INVALID_SIZE	if passed-in buflen is not valid
OS_ERROR	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 *Remote↩ Addr)

  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	*sock_id*	The server socket ID, previously bound using OS_SocketBind()
out	*connsock↩ _id*	The connection socket, a new ID that can be read/written (must not be null)
in	*Addr*	The remote address of the incoming connection (must not be null)
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

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 sock_id parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket
OS_ERR_INCORRECT_OBJ_STATE	if the socket is not bound or already connected

10.76.2.2 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	*sock↩ _id*	The socket ID
in	*Addr*	The local address to bind to (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_INVALID_POINTER	if argument is NULL
OS_ERR_INCORRECT_OBJ_STATE	if the socket is already bound
OS_ERR_INCORRECT_OBJ_TYPE	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	*sock↩ _id*	The socket ID
in	*Addr*	The remote address to connect to (must not be null)
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INCORRECT_OBJ_STATE	if the socket is already connected
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket
OS_INVALID_POINTER	if Addr 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	*sock_id*	Buffer to hold result (must not be null)
in	*sock_name*	Name of socket to find (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	is id or 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

10.76.2.5 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	*sock_id*	The socket ID
out	*sock_prop*	Buffer to hold socket information (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid semaphore
OS_INVALID_POINTER	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	*sock↩ _id*	Buffer to hold the non-zero OSAL ID (must not be null)
in	*Domain*	The domain / address family of the socket (INET or INET6, etc)
in	*Type*	The type of the socket (STREAM or DATAGRAM)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if argument is NULL
OS_ERR_NOT_IMPLEMENTED	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	*sock_id*	The socket ID, previously bound using OS_SocketBind()
out	*buffer*	Pointer to message data receive buffer (must not be null)
in	*buflen*	The maximum length of the message data to receive (must not be zero)
out	*RemoteAddr*	Buffer to store the remote network address (may be NULL)
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

Returns

Count of actual bytes received or error status, see OSAL Return Code Defines

Return values

OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_SIZE	if passed-in buflen is not valid
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket

10.76.2.8 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	*sock_id*	The socket ID, which must be of the datagram type
in	*buffer*	Pointer to message data to send (must not be null)
in	*buflen*	The length of the message data to send (must not be zero)
in	*RemoteAddr*	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

OS_INVALID_POINTER	if argument is NULL
OS_ERR_INVALID_SIZE	if passed-in buflen is not valid
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket

10.76.2.9 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	*sock↩ _id*	The socket ID
in	*Mode*	Whether to shutdown reading, writing, or both.

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the sock_id parameter is not valid
OS_ERR_INVALID_ARGUMENT	if the Mode argument is not one of the valid options
OS_ERR_INCORRECT_OBJ_TYPE	if the handle is not a socket
OS_ERR_INCORRECT_OBJ_STATE	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	*task_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*task_name*	the name of the new resource to create (must not be null)
in	*function_pointer*	the entry point of the new task (must not be null)
in	*stack_pointer*	pointer to the stack for the task, or NULL to allocate a stack from the system memory heap
in	*stack_size*	the size of the stack (must not be zero)
in	*priority*	initial priority of the new task
in	*flags*	initial options for the new task

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if any of the necessary pointers are NULL
OS_ERR_INVALID_SIZE	if the stack_size argument is zero
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_INVALID_PRIORITY	if the priority is bad (return value only verified in coverage test)
OS_ERR_NO_FREE_IDS	if there can be no more tasks created
OS_ERR_NAME_TAKEN	if the name specified is already used by a task
OS_ERROR	if an unspecified/other error occurs (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	*millisecond*	Amount of time to delay

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERROR	if an unspecified/other error occurs (return value only verified in coverage test)

Referenced by FM_ChildConcatCmd(), FM_ChildFileInfoCmd(), and FM_ChildSleepStat().

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	*task_id*	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 given to it is invalid
OS_ERROR	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	*task_id*	The buffer where the task id output is stored (must not be null)
in	*sysdata*	Pointer to the system-provided identification data
in	*sysdata_size*	Size of the system-provided identification data

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution. (return value only verified in coverage test)
OS_INVALID_POINTER	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	*task_id*	will be set to the ID of the existing resource
in	*task_name*	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	if the pointers passed in are NULL
OS_ERR_NAME_TOO_LONG	name length including null terminator greater than OS_MAX_API_NAME
OS_ERR_NAME_NOT_FOUND	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	*task_id*	The object ID to operate on
out	*task_prop*	The property object buffer to fill (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the ID passed to it is invalid
OS_INVALID_POINTER	if the task_prop pointer is NULL

Referenced by LoadOpenFileData().

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_Task↩ Delete 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	*function_pointer*	function to be called when task exits

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_ERR_INVALID_ID	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	*task_id*	The object ID to operate on
in	*new_priority*	Set the new priority

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the ID passed to it is invalid
OS_ERR_INVALID_PRIORITY	if the priority is greater than the max allowed (return value only verified in coverage test)
OS_ERROR	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	*timebase_id*	will be set to the non-zero ID of the newly-created resource (must not be null)
in	*timebase_name*	The name of the time base (must not be null)
in	*external_sync*	A synchronization function for BSP hardware-based timer ticks

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_NAME_TAKEN	if the name specified is already used
OS_ERR_NO_FREE_IDS	if there can be no more timebase resources created
OS_ERR_INCORRECT_OBJ_STATE	if called from timer/timebase context
OS_ERR_NAME_TOO_LONG	if the timebase_name is too long
OS_INVALID_POINTER	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	*timebase↩ _id*	The timebase resource to delete

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timebase
OS_ERR_INCORRECT_OBJ_STATE	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	*timebase_id*	The timebase to operate on
out	*freerun_val*	Buffer to store the free run counter (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timebase
OS_INVALID_POINTER	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	*timebase_id*	will be set to the non-zero ID of the matching resource (must not be null)
in	*timebase_name*	The name of the timebase resource to find (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_INVALID_POINTER	if timebase_id or timebase_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
OS_ERR_INCORRECT_OBJ_STATE	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	*timebase_id*	The timebase resource ID
out	*timebase_prop*	Buffer to store timebase properties (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timebase
OS_INVALID_POINTER	if the timebase_prop pointer is null
OS_ERR_INCORRECT_OBJ_STATE	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	*timebase_id*	The timebase resource to configure
in	*start_time*	The amount of delay for the first tick, in microseconds.
in	*interval_time*	The amount of delay between ticks, in microseconds.

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timebase
OS_ERR_INCORRECT_OBJ_STATE	if called from timer/timebase context
OS_TIMER_ERR_INVALID_ARGS	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_Timer↩ Callback_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	*timer_id*	Will be set to the non-zero resource ID of the timer object (must not be null)
in	*timer_name*	Name of the timer object (must not be null)
in	*timebase↩ _id*	The time base resource to use as a reference
in	*callback_ptr*	Application-provided function to invoke (must not be null)
in	*callback_arg*	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

out	*timer_id*	Will be set to the non-zero resource ID of the timer object (must not be null)
in	*timer_name*	Name of the timer object (must not be null)
out	*clock_accuracy*	Expected precision of the timer, in microseconds. This is the underlying tick value rounded to the nearest microsecond integer. (must not be null)
in	*callback_ptr*	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	*timer↩ _id*	The timer ID to operate on

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the timer_id is invalid.
OS_TIMER_ERR_INTERNAL	if there was a problem deleting the timer in the host OS (return value only verified in coverage test)
OS_ERR_INCORRECT_OBJ_STATE	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	*timer_id*	Will be set to the timer ID corresponding to the name (must not be null)
in	*timer_name*	The timer name to find (must not be null)

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.

Return values

OS_INVALID_POINTER	if timer_id or timer_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
OS_ERR_INCORRECT_OBJ_STATE	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	*timer_id*	The timer ID to operate on
out	*timer_prop*	Buffer containing timer properties (must not be null) • creator: the OS task ID of the task that created this timer • name: the string name of the timer • start_time: the start time in microseconds, if any • interval_time: the interval time in microseconds, if any • accuracy: the accuracy of the timer in microseconds

Returns

Execution status, see OSAL Return Code Defines

Return values

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the id passed in is not a valid timer
OS_INVALID_POINTER	if the timer_prop pointer is null
OS_ERR_INCORRECT_OBJ_STATE	if called from timer/timebase context

10.79.2.6 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	*timer_id*	The timer ID to operate on
in	*start_time*	Time in microseconds to the first expiration
in	*interval_time*	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

OS_SUCCESS	Successful execution.
OS_ERR_INVALID_ID	if the timer_id is not valid.
OS_TIMER_ERR_INTERNAL	if there was an error programming the OS timer (return value only verified in coverage test)
OS_ERR_INCORRECT_OBJ_STATE	if called from timer/timebase context
OS_TIMER_ERR_INVALID_ARGS	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_ResourceId_t ResourceId

  Application or Library ID for this resource.
- uint32 Type

  The type of App: CORE or EXTERNAL.
- char Name [CFE_MISSION_MAX_API_LEN]

  The Registered Name of the Application.
- char EntryPoint [CFE_MISSION_MAX_API_LEN]

  The Entry Point label for the Application.
- char FileName [CFE_MISSION_MAX_PATH_LEN]

  The Filename of the file containing the Application.
- CFE_ES_MemOffset_t StackSize

  The Stack Size of the Application.
- uint32 AddressesAreValid

  Indicates that the Code, Data, and BSS addresses/sizes are valid.
- CFE_ES_MemAddress_t CodeAddress

  The Address of the Application Code Segment.
- CFE_ES_MemOffset_t CodeSize

  The Code Size of the Application.
- CFE_ES_MemAddress_t DataAddress

  The Address of the Application Data Segment.
- CFE_ES_MemOffset_t DataSize

  The Data Size of the Application.
- CFE_ES_MemAddress_t BSSAddress

  The Address of the Application BSS Segment.
- CFE_ES_MemOffset_t BSSSize

  The BSS Size of the Application.

- CFE_ES_MemAddress_t StartAddress

  The Start Address of the Application.
- CFE_ES_ExceptionAction_Enum_t ExceptionAction

  What should occur if Application has an exception (Restart Application OR Restart Processor)
- CFE_ES_TaskPriority_Atom_t Priority

  The Priority of the Application.
- CFE_ES_TaskId_t MainTaskId

  The Application's Main Task ID.
- uint32 ExecutionCounter

  The Application's Main Task Execution Counter.
- char MainTaskName [CFE_MISSION_MAX_API_LEN]

  The Application's Main Task ID.
- uint32 NumOfChildTasks

  Number of Child tasks for an App.

11.3.1 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.

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 <cfe_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/cfe_es_msg.h

11.16 CFE_ES_HousekeepingTlm_Payload Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CommandCounter

  The ES Application Command Counter.
- uint8 CommandErrorCounter

  The ES Application Command Error Counter.
- uint16 CFECoreChecksum

  Checksum of cFE Core Code.
- uint8 CFEMajorVersion

  Major Version Number of cFE.
- uint8 CFEMinorVersion

  Minor Version Number of cFE.
- uint8 CFERevision

  Sub-Minor Version Number of cFE.
- uint8 CFEMissionRevision

  Mission Version Number of cFE.
- uint8 OSALMajorVersion

  OS Abstraction Layer Major Version Number.
- uint8 OSALMinorVersion

  OS Abstraction Layer Minor Version Number.
- uint8 OSALRevision

  OS Abstraction Layer Revision Number.
- uint8 OSALMissionRevision

  OS Abstraction Layer MissionRevision Number.
- uint8 PSPMajorVersion

  Platform Support Package Major Version Number.
- uint8 PSPMinorVersion

  Platform Support Package Minor Version Number.
- uint8 PSPRevision

  Platform Support Package Revision Number.
- uint8 PSPMissionRevision

  Platform Support Package MissionRevision Number.
- CFE_ES_MemOffset_t SysLogBytesUsed

  Total number of bytes used in system log.
- CFE_ES_MemOffset_t SysLogSize

  Total size of the system log.
- uint32 SysLogEntries

  Number of entries in the system log.
- uint32 SysLogMode

  Write/Overwrite Mode.
- uint32 ERLogIndex

  Current index of the ER Log (wraps around)
- uint32 ERLogEntries

  Number of entries made in the ER Log since the power on.
- uint32 RegisteredCoreApps

  Number of Applications registered with ES.
- uint32 RegisteredExternalApps

Number of Applications registered with ES.

- uint32 RegisteredTasks

  Number of Tasks (main AND child tasks) registered with ES.
- uint32 RegisteredLibs

  Number of Libraries registered with ES.
- uint32 ResetType

  Reset type (PROCESSOR or POWERON)
- uint32 ResetSubtype

  Reset Sub Type.
- uint32 ProcessorResets

  Number of processor resets since last power on.
- uint32 MaxProcessorResets

  Max processor resets before a power on is done.
- uint32 BootSource

  Boot source (as provided from BSP)
- uint32 PerfState

  Current state of Performance Analyzer.
- uint32 PerfMode

  Current mode of Performance Analyzer.
- uint32 PerfTriggerCount

  Number of Times Performance Analyzer has Triggered.
- uint32 PerfFilterMask [CFE_MISSION_ES_PERF_MAX_IDS/32]

  Current Setting of Performance Analyzer Filter Masks.
- uint32 PerfTriggerMask [CFE_MISSION_ES_PERF_MAX_IDS/32]

  Current Setting of Performance Analyzer Trigger Masks.
- uint32 PerfDataStart

  Identifies First Stored Entry in Performance Analyzer Log.
- uint32 PerfDataEnd

  Identifies Last Stored Entry in Performance Analyzer Log.
- uint32 PerfDataCount

  Number of Entries Put Into the Performance Analyzer Log.
- uint32 PerfDataToWrite

  Number of Performance Analyzer Log Entries Left to be Written to Log Dump File.
- CFE_ES_MemOffset_t HeapBytesFree

  Number of free bytes remaining in the OS heap.
- CFE_ES_MemOffset_t HeapBlocksFree

  Number of free blocks remaining in the OS heap.
- CFE_ES_MemOffset_t HeapMaxBlockSize

  Number of bytes in the largest free block.

11.16.1 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_BlksREQ

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 AppInfo

`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 <cfe_es_api_typedefs.h>
```

Data Fields

- void * Ptr

  Aligned pointer.
- long long int LongInt

  Aligned Long Integer.
- long double LongDouble

  Aligned Long Double.

11.24.1 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 <cfe_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/cfe_es_msg.h

11.30 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference

Send Memory Pool Statistics Command Payload.

```
#include <cfe_es_msg.h>
```

Data Fields

- char Application [CFE_MISSION_MAX_API_LEN]
  - *RESERVED - should be all zeroes*
- CFE_ES_MemHandle_t PoolHandle

  Handle of Pool whose statistics are to be telemetered.

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/cfe_es_msg.h

11.31 CFE_ES_SetMaxPRCountCmd Struct Reference

Set Maximum Processor Reset Count Command.

```
#include <cfe_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/cfe_es_msg.h

11.32 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference

Set Maximum Processor Reset Count Command Payload.

```
#include <cfe_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/cfe_es_msg.h

11.33 CFE_ES_SetPerfFilterMaskCmd Struct Reference

Set Performance Analyzer Filter Mask Command.

```
#include <cfe_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_RE↩START=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_R↩ESTART=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 AppId

  Parent Application ID.
- char AppName [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.

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 <cfe_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 <cfe_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 <cfe_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 cfe_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 cfe_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 cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.50 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference

Generic App Name and Event ID Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]

  Application name to use in the command.
- uint16 EventID

  Event ID to use in the command.

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/cfe_evs_msg.h

11.51 CFE_EVS_AppNameEventIDMaskCmd Struct Reference

Generic App Name, Event ID, Mask Command.

```
#include <cfe_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 cfe_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 cfe_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 cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/fsw/inc/cfe_evs_msg.h

11.52 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference

Generic App Name, Event ID, Mask Command Payload.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]

  Application name to use in the command.
- uint16 EventID

  Event ID to use in the command.
- uint16 Mask

  Mask to use in the command.

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 AppID

```
CFE_ES_AppId_t CFE_EVS_AppTlmData::AppID
```

Numerical application identifier.

Telemetry Mnemonic(s) $sc_$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].APPID

Definition at line 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].APPMSGSENTC

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_HousekeepingTlm 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.

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.

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 <cfe_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 <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]
  Application name.
- uint16 EventID
  Numerical event identifier.
- uint16 EventType
  Numerical event type identifier.
- uint32 SpacecraftID
  Spacecraft identifier.
- uint32 ProcessorID
  Numerical processor identifier.

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.

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.

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_WriteLogDataFileCmd 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_WriteLogDataFileCmd::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_WriteLogDataFileCmd::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 <cfe_fs_extern_typedefs.h>
```

Data Fields

- uint32 ContentType

  Identifies the content type (='cFE1'=0x63464531)
- uint32 SubType

  Type of `ContentType`, if necessary.
- uint32 Length

  Length of 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.

Referenced by FM_ChildDirListFileInit().

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.

Referenced by FM_ChildDirListFileInit().

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 GetPipeIdByNameErrorCounter

  Count of errors in get pipe id by name API.
- uint8 Spare2Align [1]

  Spare bytes to ensure alignment.
- uint16 PipeOverflowErrorCounter

  Count of pipe overflow errors.
- uint16 MsgLimitErrorCounter

  Count of msg id to pipe errors.
- CFE_ES_MemHandle_t MemPoolHandle

  Handle to SB's Memory Pool.
- uint32 MemInUse

  Memory in use.
- uint32 UnmarkedMem

  cfg param CFE_PLATFORM_SB_BUF_MEMORY_BYTES minus Peak Memory in use

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 FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_MoveFileCmd(), FM_NoopCmd(), FM_ProcessCmd(), FM_ProcessPkt(), FM_RenameFileCmd(), FM_ResetCountersCmd(), FM_SetPermissionsCmd(), and FM_SetTableStateCmd().

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_RouteId_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_PipeId_t PipeId

  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]
- char AppName [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 PipeId

```
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_PipeId_t Pipe

  Pipe ID of route to be enabled or disabled CFE_SB_PipeId_t.
- uint8 Spare

  Spare byte to make command even number of bytes.

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 <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_MsgId_t MsgId

  Message Id portion of the route.
- CFE_SB_PipeId_t PipeId

  Pipe Id portion of the route.
- uint8 State

  Route Enabled or Disabled.
- uint16 MsgCnt

  Number of msgs with this MsgId sent to this PipeId.
- char AppName [CFE_MISSION_MAX_API_LEN]

  Pipe Depth Statistics.
- char PipeName [CFE_MISSION_MAX_API_LEN]

  Pipe Depth Statistics.

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 <cfe_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 <cfe_sb_msg.h>
```

Data Fields

- uint8 SubType

  Subscription or Unsubscription.
- CFE_SB_MsgId_t MsgId

  MsgId subscribed or unsubscribe to.
- CFE_SB_Qos_t Qos

  Quality of Service, used only for interprocessor communication.
- CFE_SB_PipeId_t Pipe

  Destination pipe id to send above msg id.

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

Number of current subscriptions.

- uint32 PeakSubscriptionsInUse

  Peak number of subscriptions.
- uint32 MaxSubscriptionsAllowed

  product of CFE_PLATFORM_SB_MAX_MSG_IDS and CFE_PLATFORM_SB_MAX_DEST_PER_PKT
- uint32 SBBuffersInUse

  Number of SB message buffers currently in use.
- uint32 PeakSBBuffersInUse

  Max number of SB message buffers in use.
- uint32 MaxPipeDepthAllowed

  Maximum allowed pipe depth.
- CFE_SB_PipeDepthStats_t PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]

  Pipe Depth Statistics CFE_SB_PipeDepthStats_t.

11.90.1 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_SMPPIU

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_AllSubscriptionsTlm_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 <cfe_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/cfe_sb_msg.h

11.93 CFE_SB_WriteFileInfoCmd_Payload Struct Reference

Write File Info Command Payload.

```
#include <cfe_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/cfe_sb_msg.h

11.94 CFE_TBL_AbortLoadCmd Struct Reference

Abort Load Command.

```
#include <cfe_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 <cfe_tbl_filedef.h>
```

Data Fields

- char ObjectName [64]

  Name of instantiated variable that contains desired table image.
- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of Table as defined onboard.
- char Description [CFE_FS_HDR_DESC_MAX_LEN]

  Description of table image that is included in cFE File Header.
- char TgtFilename [CFE_MISSION_MAX_FILE_LEN]

  Default filename to be used for output of elf2cfetbl utility.
- uint32 ObjectSize

  Size, in bytes, of instantiated object.

11.105.1 Detailed Description

Definition at line 56 of file cfe_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 cfe_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 cfe_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

Indicator of whether table buffer validated was 0=Inactive, 1=Active.

- char LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of last table validated.
- uint8 SuccessValCounter

  Total number of successful table validations.
- uint8 FailedValCounter

  Total number of unsuccessful table validations.
- uint8 NumValRequests

  Number of times Table Services has requested validations from Apps.
- uint8 NumFreeSharedBufs

  Number of free Shared Working Buffers.
- uint8 ByteAlignPad1

  Spare byte to ensure longword alignment.
- CFE_ES_MemHandle_t MemPoolHandle

  Handle to TBL's memory pool.
- CFE_TIME_SysTime_t LastUpdateTime

  Time of last table update.
- char LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of the last table updated.
- char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]

  Path and Name of last table image file loaded.
- char LastFileDumped [CFE_MISSION_MAX_PATH_LEN]

  Path and Name of last file dumped to.
- char LastTableLoaded [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of the last table loaded.

11.107.1 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 <cfe_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 <cfe_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_Table↩ RegistryTlm_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 <cfe_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/cfe_tbl_msg.h

11.119 CFE_TBL_ValidateCmd_Payload Struct Reference

Validate 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 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 <cfe_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/cfe_time_msg.h

11.121 CFE_TIME_DiagnosticTlm_Payload Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- CFE_TIME_SysTime_t AtToneMET

  MET at time of tone.
- CFE_TIME_SysTime_t AtToneSTCF

  STCF at time of tone.
- CFE_TIME_SysTime_t AtToneDelay

  Adjustment for slow tone detection.
- CFE_TIME_SysTime_t AtToneLatch

  Local clock latched at time of tone.
- int16 AtToneLeapSeconds

  Leap Seconds at time of tone.
- CFE_TIME_ClockState_Enum_t ClockStateAPI

  Clock state as per API.
- CFE_TIME_SysTime_t TimeSinceTone

  Time elapsed since the tone.
- CFE_TIME_SysTime_t CurrentLatch

  Local clock latched just "now".
- CFE_TIME_SysTime_t CurrentMET

  MET at this instant.
- CFE_TIME_SysTime_t CurrentTAI

  TAI at this instant.
- CFE_TIME_SysTime_t CurrentUTC

  UTC at this instant.
- int16 ClockSetState

  Time has been "set".
- int16 ClockFlyState

  Current fly-wheel state.
- int16 ClockSource

  Internal vs external, etc.
- int16 ClockSignal

  Primary vs redundant, etc.
- int16 ServerFlyState

  Used by clients only.
- int16 Forced2Fly

  Commanded into fly-wheel.
- uint16 ClockStateFlags

  Clock State Flags.
- int16 OneTimeDirection

  One time STCF adjustment direction (Add = 1, Sub = 2)
- int16 OneHzDirection

  1Hz STCF adjustment direction
- int16 DelayDirection

  Client latency adjustment direction.
- CFE_TIME_SysTime_t OneTimeAdjust

  Previous one-time STCF adjustment.
- CFE_TIME_SysTime_t OneHzAdjust

Current 1Hz STCF adjustment.

- CFE_TIME_SysTime_t ToneSignalLatch

  Local Clock latched at most recent tone signal.
- CFE_TIME_SysTime_t ToneDataLatch

  Local Clock latched at arrival of tone data.
- uint32 ToneMatchCounter

  Tone signal / data verification count.
- uint32 ToneMatchErrorCounter

  Tone signal / data verification error count.
- uint32 ToneSignalCounter

  Tone signal detected SB message count.
- uint32 ToneDataCounter

  Time at the tone data SB message count.
- uint32 ToneIntCounter

  Tone signal ISR execution count.
- uint32 ToneIntErrorCounter

  Tone signal ISR error count.
- uint32 ToneTaskCounter

  Tone task execution count.
- uint32 VersionCounter

  Count of mods to time at tone reference data (version)
- uint32 LocalIntCounter

  Local 1Hz ISR execution count.
- uint32 LocalTaskCounter

  Local 1Hz task execution count.
- uint32 VirtualMET

  Software MET.
- uint32 MinElapsed

  Min tone signal / data pkt arrival window (Sub-seconds)
- uint32 MaxElapsed

  Max tone signal / data pkt arrival window (Sub-seconds)
- CFE_TIME_SysTime_t MaxLocalClock

  Max local clock value before rollover.
- uint32 ToneOverLimit

  Max between tone signal interrupts.
- uint32 ToneUnderLimit

  Min between tone signal interrupts.
- uint32 DataStoreStatus

  Data Store status (preserved across processor reset)

11.121.1 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_DLatentS, $sc_$cpu_TIME_DLatentSs

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_DTValidS, $sc_$cpu_TIME_DTValidSs

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_DLeapS

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_DTMETSs

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_DFlagAdjd, $sc_$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_DLatentDir

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_DTsISRERR

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_DLogicalMET

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_DAPIState

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_FlagAdjd, $sc_$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 <cfe_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 <cfe_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_VAL↩ID=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 FM_ChildQueueEntry_t Struct Reference

Child Task Interface command queue entry structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_FcnCode_t CommandCode

  Command code - identifies the command.
- uint32 DirListOffset

  Starting entry for dir list commands.
- uint32 FileInfoState

  File info state.
- uint32 FileInfoSize

  File info size.
- uint32 FileInfoTime

  File info time.
- uint32 FileInfoCRC

  File info CRC method.
- char Source1 [OS_MAX_PATH_LEN]

  First source file or directory name command argument.
- char Source2 [OS_MAX_PATH_LEN]

  Second source filename command argument.
- char Target [OS_MAX_PATH_LEN]

  Target filename command argument.
- uint8 GetSizeTimeMode

  Whether to invoke stat call for size and time (CPU intensive)
- uint32 Mode

  File Mode.

11.140.1 Detailed Description

Child Task Interface command queue entry structure.

Definition at line 497 of file fm_msg.h.

11.140.2 Field Documentation

11.140.2.1 CommandCode

```
CFE_MSG_FcnCode_t FM_ChildQueueEntry_t::CommandCode
```

Command code - identifies the command.

Definition at line 499 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildMoveCmd(), FM_ChildProcess(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

11.140.2.2 DirListOffset

```
uint32 FM_ChildQueueEntry_t::DirListOffset
```

Starting entry for dir list commands.

Definition at line 500 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd(), and FM_GetDirListPktCmd().

11.140.2.3 FileInfoCRC

```
uint32 FM_ChildQueueEntry_t::FileInfoCRC
```

File info CRC method.

Definition at line 504 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd(), and FM_GetFileInfoCmd().

11.140.2.4 FileInfoSize

```
uint32 FM_ChildQueueEntry_t::FileInfoSize
```

File info size.

Definition at line 502 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd(), and FM_GetFileInfoCmd().

11.140.2.5 FileInfoState

```
uint32 FM_ChildQueueEntry_t::FileInfoState
```

File info state.

Definition at line 501 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd(), and FM_GetFileInfoCmd().

11.140.2.6 FileInfoTime

```
uint32 FM_ChildQueueEntry_t::FileInfoTime
```

File info time.

Definition at line 503 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd(), and FM_GetFileInfoCmd().

11.140.2.7 GetSizeTimeMode

```
uint8 FM_ChildQueueEntry_t::GetSizeTimeMode
```

Whether to invoke stat call for size and time (CPU intensive)

Definition at line 508 of file fm_msg.h.

Referenced by FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_GetDirListFileCmd(), and FM_GetDirList↩PktCmd().

11.140.2.8 Mode

```
uint32 FM_ChildQueueEntry_t::Mode
```

File Mode.

Definition at line 509 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd(), FM_ChildSetPermissionsCmd(), FM_GetFileInfoCmd(), and FM_Set↩PermissionsCmd().

11.140.2.9 Source1

```
char FM_ChildQueueEntry_t::Source1[OS_MAX_PATH_LEN]
```

First source file or directory name command argument.

Definition at line 505 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_Child↩FileInfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_ConcatFiles↩Cmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), F↩M_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

11.140.2.10 Source2

```
char FM_ChildQueueEntry_t::Source2[OS_MAX_PATH_LEN]
```

Second source filename command argument.

Definition at line 506 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildDeleteAllCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPkt↩Cmd(), FM_ConcatFilesCmd(), FM_DeleteAllFilesCmd(), FM_GetDirListFileCmd(), and FM_GetDirListPktCmd().

11.140.2.11 Target

```
char FM_ChildQueueEntry_t::Target[OS_MAX_PATH_LEN]
```

Target filename command argument.

Definition at line 507 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildDeleteAllCmd(), FM_ChildDirListFileCmd(), F↩M_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_DeleteAllFilesCmd(), FM_GetDirListFileCmd(), FM_MoveFileCmd(), and FM_RenameFileCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.141 FM_ConcatCmd_t Struct Reference

Concatenate Files command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Source1 [OS_MAX_PATH_LEN]

  Source 1 filename.
- char Source2 [OS_MAX_PATH_LEN]

  Source 2 filename.
- char Target [OS_MAX_PATH_LEN]

  Target filename.

11.141.1 Detailed Description

Concatenate Files command packet structure.

For command details see FM_CONCAT_CC

Definition at line 166 of file fm_msg.h.

11.141.2 Field Documentation

11.141.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_ConcatCmd_t::CmdHeader
```

Command header.

Definition at line 168 of file fm_msg.h.

11.141.2.2 Source1

```
char FM_ConcatCmd_t::Source1[OS_MAX_PATH_LEN]
```

Source 1 filename.

Definition at line 170 of file fm_msg.h.

Referenced by FM_ConcatFilesCmd().

11.141.2.3 Source2

```
char FM_ConcatCmd_t::Source2[OS_MAX_PATH_LEN]
```

Source 2 filename.

Definition at line 171 of file fm_msg.h.

Referenced by FM_ConcatFilesCmd().

11.141.2.4 Target

```
char FM_ConcatCmd_t::Target[OS_MAX_PATH_LEN]
```

Target filename.

Definition at line 172 of file fm_msg.h.

Referenced by FM_ConcatFilesCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.142 FM_CopyFileCmd_t Struct Reference

Copy File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- uint16 Overwrite

  Allow overwrite.
- char Source [OS_MAX_PATH_LEN]

  Source filename.
- char Target [OS_MAX_PATH_LEN]

  Target filename.

11.142.1 Detailed Description

Copy File command packet structure.

For command details see FM_COPY_CC

Definition at line 84 of file fm_msg.h.

11.142.2 Field Documentation

11.142.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_CopyFileCmd_t::CmdHeader
```

Command header.

Definition at line 87 of file fm_msg.h.

11.142.2.2 Overwrite

```
uint16 FM_CopyFileCmd_t::Overwrite
```

Allow overwrite.

Definition at line 89 of file fm_msg.h.

Referenced by FM_CopyFileCmd().

11.142.2.3 Source

```
char FM_CopyFileCmd_t::Source[OS_MAX_PATH_LEN]
```

Source filename.

Definition at line 90 of file fm_msg.h.

Referenced by FM_CopyFileCmd().

11.142.2.4 Target

```
char FM_CopyFileCmd_t::Target[OS_MAX_PATH_LEN]
```

Target filename.

Definition at line 91 of file fm_msg.h.

Referenced by FM_CopyFileCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.143 FM_CreateDirCmd_t Struct Reference

Create Directory command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.
- char Directory [OS_MAX_PATH_LEN]
  Directory name.

11.143.1 Detailed Description

Create Directory command packet structure.

For command details see FM_CREATE_DIR_CC

Definition at line 207 of file fm_msg.h.

11.143.2 Field Documentation

11.143.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_CreateDirCmd_t::CmdHeader
```

Command header.

Definition at line 209 of file fm_msg.h.

11.143.2.2 Directory

```
char FM_CreateDirCmd_t::Directory[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 211 of file fm_msg.h.

Referenced by FM_CreateDirectoryCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.144 FM_DecompressCmd_t Struct Reference

Decompress File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.
- char Source [OS_MAX_PATH_LEN]
  Source filename.
- char Target [OS_MAX_PATH_LEN]
  Target filename.

11.144.1 Detailed Description

Decompress File command packet structure.

For command details see FM_DECOMPRESS_CC

Definition at line 153 of file fm_msg.h.

11.144.2 Field Documentation

11.144.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_DecompressCmd_t::CmdHeader
```

Command header.

Definition at line 155 of file fm_msg.h.

11.144.2.2 Source

```
char FM_DecompressCmd_t::Source[OS_MAX_PATH_LEN]
```

Source filename.

Definition at line 156 of file fm_msg.h.

Referenced by FM_DeleteAllFilesCmd().

11.144.2.3 Target

```
char FM_DecompressCmd_t::Target[OS_MAX_PATH_LEN]
```

Target filename.

Definition at line 157 of file fm_msg.h.

Referenced by FM_DeleteAllFilesCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.145 FM_DeleteAllCmd_t Struct Reference

Delete All command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Directory [OS_MAX_PATH_LEN]

  Directory name.

11.145.1 Detailed Description

Delete All command packet structure.

For command details see FM_DELETE_ALL_CC

Definition at line 141 of file fm_msg.h.

11.145.2 Field Documentation

11.145.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_DeleteAllCmd_t::CmdHeader
```

Command header.

Definition at line 143 of file fm_msg.h.

11.145.2.2 Directory

```
char FM_DeleteAllCmd_t::Directory[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 144 of file fm_msg.h.

Referenced by FM_DeleteAllFilesCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.146 FM_DeleteDirCmd_t Struct Reference

Delete Directory command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Directory [OS_MAX_PATH_LEN]

  Directory name.

11.146.1 Detailed Description

Delete Directory command packet structure.

For command details see FM_DELETE_DIR_CC

Definition at line 220 of file fm_msg.h.

11.146.2 Field Documentation

11.146.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_DeleteDirCmd_t::CmdHeader
```

Command header.

Definition at line 222 of file fm_msg.h.

11.146.2.2 Directory

```
char FM_DeleteDirCmd_t::Directory[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 224 of file fm_msg.h.

Referenced by FM_DeleteDirectoryCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.147 FM_DeleteFileCmd_t Struct Reference

Delete File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Filename [OS_MAX_PATH_LEN]

  Delete filename.

11.147.1 Detailed Description

Delete File command packet structure.

For command details see FM_DELETE_CC

Definition at line 129 of file fm_msg.h.

11.147.2 Field Documentation

11.147.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_DeleteFileCmd_t::CmdHeader
```

Command header.

Definition at line 131 of file fm_msg.h.

11.147.2.2 Filename

```
char FM_DeleteFileCmd_t::Filename[OS_MAX_PATH_LEN]
```

Delete filename.

Definition at line 132 of file fm_msg.h.

Referenced by FM_DeleteFileCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.148 FM_DirListEntry_t Struct Reference

Get Directory Listing entry structure.

```
#include <fm_msg.h>
```

Data Fields

- char EntryName [OS_MAX_PATH_LEN]

  Directory Listing Filename.
- uint32 EntrySize

  Directory Listing File Size.
- uint32 ModifyTime

  Directory Listing File Last Modification Times.
- uint32 Mode

  Mode of the file (Permissions from OS_FILESTAT_MODE)

11.148.1 Detailed Description

Get Directory Listing entry structure.

Definition at line 317 of file fm_msg.h.

11.148.2 Field Documentation

11.148.2.1 EntryName

```
char FM_DirListEntry_t::EntryName[OS_MAX_PATH_LEN]
```

Directory Listing Filename.

Definition at line 319 of file fm_msg.h.

Referenced by FM_ChildDirListFileLoop(), and FM_ChildDirListPktCmd().

11.148.2.2 EntrySize

```
uint32 FM_DirListEntry_t::EntrySize
```

Directory Listing File Size.

Definition at line 320 of file fm_msg.h.

Referenced by FM_ChildSleepStat().

11.148.2.3 Mode

```
uint32 FM_DirListEntry_t::Mode
```

Mode of the file (Permissions from OS_FILESTAT_MODE)

Definition at line 322 of file fm_msg.h.

Referenced by FM_ChildSleepStat().

11.148.2.4 ModifyTime

```
uint32 FM_DirListEntry_t::ModifyTime
```

Directory Listing File Last Modification Times.

Definition at line 321 of file fm_msg.h.

Referenced by FM_ChildSleepStat().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.149 FM_DirListFileStats_t Struct Reference

Get Directory Listing file statistics structure.

```
#include <fm_msg.h>
```

Data Fields

- char DirName [OS_MAX_PATH_LEN]

  Directory name.
- uint32 DirEntries

  Number of entries in the directory.
- uint32 FileEntries

  Number of entries written to output file.

11.149.1 Detailed Description

Get Directory Listing file statistics structure.

Definition at line 348 of file fm_msg.h.

11.149.2 Field Documentation

11.149.2.1 DirEntries

```
uint32 FM_DirListFileStats_t::DirEntries
```

Number of entries in the directory.

Definition at line 351 of file fm_msg.h.

Referenced by FM_ChildDirListFileLoop().

11.149.2.2 DirName

```
char FM_DirListFileStats_t::DirName[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 350 of file fm_msg.h.

Referenced by FM_ChildDirListFileInit().

11.149.2.3 FileEntries

```
uint32 FM_DirListFileStats_t::FileEntries
```

Number of entries written to output file.

Definition at line 352 of file fm_msg.h.

Referenced by FM_ChildDirListFileLoop().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.150 FM_DirListPkt_t Struct Reference

Get Directory Listing telemetry packet.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_TelemetryHeader_t TlmHeader

  Telemetry Header.
- char DirName [OS_MAX_PATH_LEN]

  Directory Name.
- uint32 TotalFiles

  Number of files in the directory.
- uint32 PacketFiles

  Number of files in this packet.
- uint32 FirstFile

  Index into directory files of first packet file.
- FM_DirListEntry_t FileList [FM_DIR_LIST_PKT_ENTRIES]

  Directory listing file data.

11.150.1 Detailed Description

Get Directory Listing telemetry packet.

Definition at line 328 of file fm_msg.h.

11.150.2 Field Documentation

11.150.2.1 DirName

```
char FM_DirListPkt_t::DirName[OS_MAX_PATH_LEN]
```

Directory Name.

Definition at line 332 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.150.2.2 FileList

```
FM_DirListEntry_t FM_DirListPkt_t::FileList[FM_DIR_LIST_PKT_ENTRIES]
```

Directory listing file data.

Definition at line 336 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.150.2.3 FirstFile

```
uint32 FM_DirListPkt_t::FirstFile
```

Index into directory files of first packet file.

Definition at line 335 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.150.2.4 PacketFiles

```
uint32 FM_DirListPkt_t::PacketFiles
```

Number of files in this packet.

Definition at line 334 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.150.2.5 TlmHeader

`CFE_MSG_TelemetryHeader_t FM_DirListPkt_t::TlmHeader`

Telemetry Header.

Definition at line 330 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.150.2.6 TotalFiles

`uint32 FM_DirListPkt_t::TotalFiles`

Number of files in the directory.

Definition at line 333 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.151 FM_FileInfoPkt_t Struct Reference

Get File Info telemetry packet.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_TelemetryHeader_t TlmHeader

  Telemetry Header.
- uint8 FileStatus

  Status indicating whether the file is open or closed.
- uint8 CRC_Computed

  Flag indicating whether a CRC was computed or not.
- uint8 Spare [2]

  Structure padding.
- uint32 CRC

  CRC value if computed.
- uint32 FileSize

  File Size.
- uint32 LastModifiedTime

  Last Modification Time of File.
- uint32 Mode

  Mode of the file (Permissions)
- char Filename [OS_MAX_PATH_LEN]

  Name of File.

11.151.1 Detailed Description

Get File Info telemetry packet.

Definition at line 365 of file fm_msg.h.

11.151.2 Field Documentation

11.151.2.1 CRC

```
uint32 FM_FileInfoPkt_t::CRC
```

CRC value if computed.

Definition at line 372 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.2 CRC_Computed

```
uint8 FM_FileInfoPkt_t::CRC_Computed
```

Flag indicating whether a CRC was computed or not.

Definition at line 370 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.3 Filename

```
char FM_FileInfoPkt_t::Filename[OS_MAX_PATH_LEN]
```

Name of File.

Definition at line 376 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.4 FileSize

```
uint32 FM_FileInfoPkt_t::FileSize
```

File Size.

Definition at line 373 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.5 FileStatus

```
uint8 FM_FileInfoPkt_t::FileStatus
```

Status indicating whether the file is open or closed.

Definition at line 369 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.6 LastModifiedTime

```
uint32 FM_FileInfoPkt_t::LastModifiedTime
```

Last Modification Time of File.

Definition at line 374 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.7 Mode

```
uint32 FM_FileInfoPkt_t::Mode
```

Mode of the file (Permissions)

Definition at line 375 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.151.2.8 Spare

```
uint8 FM_FileInfoPkt_t::Spare[2]
```

Structure padding.

Definition at line 371 of file fm_msg.h.

11.151.2.9 TlmHeader

```
CFE_MSG_TelemetryHeader_t FM_FileInfoPkt_t::TlmHeader
```

Telemetry Header.

Definition at line 367 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.152 FM_FreeSpacePkt_t Struct Reference

Get Free Space telemetry packet.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_TelemetryHeader_t TlmHeader
  Telemetry Header.
- FM_FreeSpacePktEntry_t FileSys [FM_TABLE_ENTRY_COUNT]
  Array of file system free space entries.

11.152.1 Detailed Description

Get Free Space telemetry packet.

Definition at line 424 of file fm_msg.h.

11.152.2 Field Documentation

11.152.2.1 FileSys

```
FM_FreeSpacePktEntry_t FM_FreeSpacePkt_t::FileSys[FM_TABLE_ENTRY_COUNT]
```

Array of file system free space entries.

Definition at line 428 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd().

11.152.2.2 TlmHeader

```
CFE_MSG_TelemetryHeader_t FM_FreeSpacePkt_t::TlmHeader
```

Telemetry Header.

Definition at line 426 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.153 FM_FreeSpacePktEntry_t Struct Reference

Get Free Space list entry structure.

```
#include <fm_msg.h>
```

Data Fields

- osal_blockcount_t FreeSpace

  Blocks free in filesystem.
- char Name [OS_MAX_PATH_LEN]

  File system name.

11.153.1 Detailed Description

Get Free Space list entry structure.

Definition at line 415 of file fm_msg.h.

11.153.2 Field Documentation

11.153.2.1 FreeSpace

```
osal_blockcount_t FM_FreeSpacePktEntry_t::FreeSpace
```

Blocks free in filesystem.

Definition at line 417 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd().

11.153.2.2 Name

```
char FM_FreeSpacePktEntry_t::Name[OS_MAX_PATH_LEN]
```

File system name.

Definition at line 418 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.154 FM_FreeSpaceTable_t Struct Reference

Get Free Space table definition.

```
#include <fm_msg.h>
```

Data Fields

- FM_TableEntry_t FileSys [FM_TABLE_ENTRY_COUNT]
  One entry for each file system.

11.154.1 Detailed Description

Get Free Space table definition.

Definition at line 482 of file fm_msg.h.

11.154.2 Field Documentation

11.154.2.1 FileSys

```
FM_TableEntry_t FM_FreeSpaceTable_t::FileSys[FM_TABLE_ENTRY_COUNT]
```

One entry for each file system.

Definition at line 484 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd(), FM_SetTableStateCmd(), and FM_ValidateTable().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.155 FM_GetDirFileCmd_t Struct Reference

Get DIR List to File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Directory [OS_MAX_PATH_LEN]

  Directory name.
- char Filename [OS_MAX_PATH_LEN]

  Filename.
- uint8 GetSizeTimeMode

  Option to query size, time, and mode of files (CPU intensive)
- uint8 Spare01 [3]

  Padding to 32 bit boundary.

11.155.1 Detailed Description

Get DIR List to File command packet structure.

For command details see FM_GET_DIR_FILE_CC

Definition at line 233 of file fm_msg.h.

11.155.2 Field Documentation

11.155.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_GetDirFileCmd_t::CmdHeader
```

Command header.

Definition at line 235 of file fm_msg.h.

11.155.2.2 Directory

```
char FM_GetDirFileCmd_t::Directory[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 237 of file fm_msg.h.

Referenced by FM_GetDirListFileCmd().

11.155.2.3 Filename

```
char FM_GetDirFileCmd_t::Filename[OS_MAX_PATH_LEN]
```

Filename.

Definition at line 238 of file fm_msg.h.

Referenced by FM_GetDirListFileCmd().

11.155.2.4 GetSizeTimeMode

```
uint8 FM_GetDirFileCmd_t::GetSizeTimeMode
```

Option to query size, time, and mode of files (CPU intensive)

Definition at line 239 of file fm_msg.h.

Referenced by FM_GetDirListFileCmd().

11.155.2.5 Spare01

```
uint8 FM_GetDirFileCmd_t::Spare01[3]
```

Padding to 32 bit boundary.

Definition at line 240 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.156 FM_GetDirPktCmd_t Struct Reference

Get DIR List to Packet command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Directory [OS_MAX_PATH_LEN]

  Directory name.
- uint32 DirListOffset

  Index of 1st dir entry to put in packet.
- uint8 GetSizeTimeMode

  Option to query size, time, and mode of files (CPU intensive)
- uint8 Spare01 [3]

  Padding to 32 bit boundary.

11.156.1 Detailed Description

Get DIR List to Packet command packet structure.

For command details see FM_GET_DIR_PKT_CC

Definition at line 249 of file fm_msg.h.

11.156.2 Field Documentation

11.156.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_GetDirPktCmd_t::CmdHeader
```

Command header.

Definition at line 252 of file fm_msg.h.

11.156.2.2 Directory

```
char FM_GetDirPktCmd_t::Directory[OS_MAX_PATH_LEN]
```

Directory name.

Definition at line 254 of file fm_msg.h.

Referenced by FM_GetDirListPktCmd().

11.156.2.3 DirListOffset

```
uint32 FM_GetDirPktCmd_t::DirListOffset
```

Index of 1st dir entry to put in packet.

Definition at line 255 of file fm_msg.h.

Referenced by FM_GetDirListPktCmd().

11.156.2.4 GetSizeTimeMode

```
uint8 FM_GetDirPktCmd_t::GetSizeTimeMode
```

Option to query size, time, and mode of files (CPU intensive)

Definition at line 256 of file fm_msg.h.

Referenced by FM_GetDirListPktCmd().

11.156.2.5 Spare01

`uint8 FM_GetDirPktCmd_t::Spare01[3]`

Padding to 32 bit boundary.

Definition at line 257 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.157 FM_GetFileInfoCmd_t Struct Reference

Get File Info command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Filename [OS_MAX_PATH_LEN]

  Filename.
- uint32 FileInfoCRC

  File info CRC method.

11.157.1 Detailed Description

Get File Info command packet structure.

For command details see FM_GET_FILE_INFO_CC

Definition at line 181 of file fm_msg.h.

11.157.2 Field Documentation

11.157.2.1 CmdHeader

`CFE_MSG_CommandHeader_t FM_GetFileInfoCmd_t::CmdHeader`

Command header.

Definition at line 183 of file fm_msg.h.

11.157.2.2 FileInfoCRC

```
uint32 FM_GetFileInfoCmd_t::FileInfoCRC
```

File info CRC method.

Definition at line 186 of file fm_msg.h.

Referenced by FM_GetFileInfoCmd().

11.157.2.3 Filename

```
char FM_GetFileInfoCmd_t::Filename[OS_MAX_PATH_LEN]
```

Filename.

Definition at line 185 of file fm_msg.h.

Referenced by FM_GetFileInfoCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.158 FM_GetFreeSpaceCmd_t Struct Reference

Get Free Space command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.

11.158.1 Detailed Description

Get Free Space command packet structure.

For command details see FM_GET_FREE_SPACE_CC

Definition at line 266 of file fm_msg.h.

11.158.2 Field Documentation

11.158.2.1 CmdHeader

`CFE_MSG_CommandHeader_t FM_GetFreeSpaceCmd_t::CmdHeader`

Command header.

Definition at line 269 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.159 FM_GetOpenFilesCmd_t Struct Reference

Get Open Files command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.

11.159.1 Detailed Description

Get Open Files command packet structure.

For command details see FM_GET_OPEN_FILES_CC

Definition at line 195 of file fm_msg.h.

11.159.2 Field Documentation

11.159.2.1 CmdHeader

`CFE_MSG_CommandHeader_t FM_GetOpenFilesCmd_t::CmdHeader`

Command header.

Definition at line 198 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.160 FM_GlobalData_t Struct Reference

Application global data structure.

```
#include <fm_msg.h>
```

Data Fields

- FM_FreeSpaceTable_t * FreeSpaceTablePtr

  File System Table Pointer.
- CFE_TBL_Handle_t FreeSpaceTableHandle

  File System Table Handle.
- CFE_SB_PipeId_t CmdPipe

  cFE software bus command pipe
- CFE_ES_TaskId_t ChildTaskID

  Child task ID.
- osal_id_t ChildSemaphore

  Child task wakeup counting semaphore.
- osal_id_t ChildQueueCountSem

  Child queue counter mutex semaphore.
- uint8 ChildCmdCounter

  Child task command success counter.
- uint8 ChildCmdErrCounter

  Child task command error counter.
- uint8 ChildCmdWarnCounter

  Child task command warning counter.
- uint8 ChildWriteIndex

  Array index for next write to command args.
- uint8 ChildReadIndex

  Array index for next read from command args.
- uint8 ChildQueueCount

  Number of pending commands in queue.
- uint8 CommandCounter

  Application command success counter.

- uint8 CommandErrCounter

  Application command error counter.
- uint8 Spare8a

  Placeholder for unused command warning counter.
- uint8 ChildCurrentCC

  Command code currently executing.
- uint8 ChildPreviousCC

  Command code previously executed.
- uint8 Spare8b

  Structure alignment spare.
- uint32 FileStatTime

  Modify time from most recent OS_stat.
- uint32 FileStatSize

  File size from most recent OS_stat.
- uint32 FileStatMode

  File mode from most recent OS_stat (OS_FILESTAT_MODE)
- FM_DirListFileStats_t DirListFileStats

  Get dir list to file statistics structure.
- FM_DirListPkt_t DirListPkt

  Get dir list to packet telemetry packet.
- FM_FreeSpacePkt_t FreeSpacePkt

  Get free space telemetry packet.
- FM_FileInfoPkt_t FileInfoPkt

  Get file info telemetry packet.
- FM_OpenFilesPkt_t OpenFilesPkt

  Get open files telemetry packet.
- FM_HousekeepingPkt_t HousekeepingPkt

  Application housekeeping telemetry packet.
- char ChildBuffer [FM_CHILD_FILE_BLOCK_SIZE]

  Child task file I/O buffer.
- FM_ChildQueueEntry_t ChildQueue [FM_CHILD_QUEUE_DEPTH]

  Child task command queue.

11.160.1 Detailed Description

Application global data structure.

Definition at line 521 of file fm_msg.h.

11.160.2 Field Documentation

11.160.2.1 ChildBuffer

```
char FM_GlobalData_t::ChildBuffer[FM_CHILD_FILE_BLOCK_SIZE]
```

Child task file I/O buffer.

Definition at line 564 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), and FM_ChildFileInfoCmd().

11.160.2.2 ChildCmdCounter

```
uint8 FM_GlobalData_t::ChildCmdCounter
```

Child task command success counter.

Definition at line 532 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_ReportHK(), and FM_ResetCountersCmd().

11.160.2.3 ChildCmdErrCounter

```
uint8 FM_GlobalData_t::ChildCmdErrCounter
```

Child task command error counter.

Definition at line 533 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildLoop(), FM_ChildMoveCmd(), FM_ChildProcess(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_ReportHK(), and FM_ResetCountersCmd().

11.160.2.4 ChildCmdWarnCounter

```
uint8 FM_GlobalData_t::ChildCmdWarnCounter
```

Child task command warning counter.

Definition at line 534 of file fm_msg.h.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ReportHK(), and FM_ResetCountersCmd().

11.160.2.5 ChildCurrentCC

```
uint8 FM_GlobalData_t::ChildCurrentCC
```

Command code currently executing.

Definition at line 544 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), F↩M_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_ChildFile↩InfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), and FM_ReportHK().

11.160.2.6 ChildPreviousCC

```
uint8 FM_GlobalData_t::ChildPreviousCC
```

Command code previously executed.

Definition at line 545 of file fm_msg.h.

Referenced by FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), F↩M_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_ChildFile↩InfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), and FM_ReportHK().

11.160.2.7 ChildQueue

```
FM_ChildQueueEntry_t FM_GlobalData_t::ChildQueue[FM_CHILD_QUEUE_DEPTH]
```

Child task command queue.

Definition at line 566 of file fm_msg.h.

Referenced by FM_ChildProcess(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_↩DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPkt↩Cmd(), FM_GetFileInfoCmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), FM_SetPermissionsCmd(), and FM_↩VerifyChildTask().

11.160.2.8 ChildQueueCount

```
uint8 FM_GlobalData_t::ChildQueueCount
```

Number of pending commands in queue.

Definition at line 538 of file fm_msg.h.

Referenced by FM_ChildLoop(), FM_ChildProcess(), FM_InvokeChildTask(), FM_ReportHK(), and FM_VerifyChild↩Task().

11.160.2.9 ChildQueueCountSem

```
osal_id_t FM_GlobalData_t::ChildQueueCountSem
```

Child queue counter mutex semaphore.

Definition at line 530 of file fm_msg.h.

Referenced by FM_ChildInit(), FM_ChildProcess(), and FM_InvokeChildTask().

11.160.2.10 ChildReadIndex

```
uint8 FM_GlobalData_t::ChildReadIndex
```

Array index for next read from command args.

Definition at line 537 of file fm_msg.h.

Referenced by FM_ChildLoop(), and FM_ChildProcess().

11.160.2.11 ChildSemaphore

```
osal_id_t FM_GlobalData_t::ChildSemaphore
```

Child task wakeup counting semaphore.

Definition at line 529 of file fm_msg.h.

Referenced by FM_ChildInit(), FM_ChildLoop(), FM_ChildTask(), FM_InvokeChildTask(), and FM_VerifyChildTask().

11.160.2.12 ChildTaskID

```
CFE_ES_TaskId_t FM_GlobalData_t::ChildTaskID
```

Child task ID.

Definition at line 528 of file fm_msg.h.

Referenced by FM_ChildInit().

11.160.2.13 ChildWriteIndex

```
uint8 FM_GlobalData_t::ChildWriteIndex
```

Array index for next write to command args.

Definition at line 536 of file fm_msg.h.

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_InvokeChildTask(), FM_MoveFileCmd(), FM_RenameFileCmd(), FM_SetPermissionsCmd(), and FM_VerifyChildTask().

11.160.2.14 CmdPipe

```
CFE_SB_PipeId_t FM_GlobalData_t::CmdPipe
```

cFE software bus command pipe

Definition at line 526 of file fm_msg.h.

Referenced by FM_AppInit(), and FM_AppMain().

11.160.2.15 CommandCounter

```
uint8 FM_GlobalData_t::CommandCounter
```

Application command success counter.

Definition at line 540 of file fm_msg.h.

Referenced by FM_ProcessCmd(), FM_ReportHK(), and FM_ResetCountersCmd().

11.160.2.16 CommandErrCounter

```
uint8 FM_GlobalData_t::CommandErrCounter
```

Application command error counter.

Definition at line 541 of file fm_msg.h.

Referenced by FM_ProcessCmd(), FM_ReportHK(), and FM_ResetCountersCmd().

11.160.2.17 DirListFileStats

```
FM_DirListFileStats_t FM_GlobalData_t::DirListFileStats
```

Get dir list to file statistics structure.

Definition at line 552 of file fm_msg.h.

Referenced by FM_ChildDirListFileInit(), and FM_ChildDirListFileLoop().

11.160.2.18 DirListPkt

```
FM_DirListPkt_t FM_GlobalData_t::DirListPkt
```

Get dir list to packet telemetry packet.

Definition at line 554 of file fm_msg.h.

Referenced by FM_ChildDirListPktCmd().

11.160.2.19 FileInfoPkt

```
FM_FileInfoPkt_t FM_GlobalData_t::FileInfoPkt
```

Get file info telemetry packet.

Definition at line 558 of file fm_msg.h.

Referenced by FM_ChildFileInfoCmd().

11.160.2.20 FileStatMode

```
uint32 FM_GlobalData_t::FileStatMode
```

File mode from most recent OS_stat (OS_FILESTAT_MODE)

Definition at line 550 of file fm_msg.h.

Referenced by FM_GetFileInfoCmd(), and FM_GetFilenameState().

11.160.2.21 FileStatSize

```
uint32 FM_GlobalData_t::FileStatSize
```

File size from most recent OS_stat.

Definition at line 549 of file fm_msg.h.

Referenced by FM_GetFileInfoCmd(), and FM_GetFilenameState().

11.160.2.22 FileStatTime

```
uint32 FM_GlobalData_t::FileStatTime
```

Modify time from most recent OS_stat.

Definition at line 548 of file fm_msg.h.

Referenced by FM_GetFileInfoCmd(), and FM_GetFilenameState().

11.160.2.23 FreeSpacePkt

```
FM_FreeSpacePkt_t FM_GlobalData_t::FreeSpacePkt
```

Get free space telemetry packet.

Definition at line 556 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd().

11.160.2.24 FreeSpaceTableHandle

```
CFE_TBL_Handle_t FM_GlobalData_t::FreeSpaceTableHandle
```

File System Table Handle.

Definition at line 524 of file fm_msg.h.

Referenced by FM_AcquireTablePointers(), FM_ReleaseTablePointers(), FM_SetTableStateCmd(), and FM_TableInit().

11.160.2.25 FreeSpaceTablePtr

```
FM_FreeSpaceTable_t* FM_GlobalData_t::FreeSpaceTablePtr
```

File System Table Pointer.

Definition at line 523 of file fm_msg.h.

Referenced by FM_AcquireTablePointers(), FM_GetFreeSpaceCmd(), FM_ReleaseTablePointers(), FM_SetTable↩StateCmd(), and FM_TableInit().

11.160.2.26 HousekeepingPkt

```
FM_HousekeepingPkt_t FM_GlobalData_t::HousekeepingPkt
```

Application housekeeping telemetry packet.

Definition at line 562 of file fm_msg.h.

Referenced by FM_ReportHK().

11.160.2.27 OpenFilesPkt

```
FM_OpenFilesPkt_t FM_GlobalData_t::OpenFilesPkt
```

Get open files telemetry packet.

Definition at line 560 of file fm_msg.h.

Referenced by FM_GetOpenFilesCmd().

11.160.2.28 Spare8a

```
uint8 FM_GlobalData_t::Spare8a
```

Placeholder for unused command warning counter.

Definition at line 542 of file fm_msg.h.

11.160.2.29 Spare8b

```
uint8 FM_GlobalData_t::Spare8b
```

Structure alignment spare.

Definition at line 546 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.161 FM_HousekeepingCmd_t Struct Reference

Housekeeping Request command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.

11.161.1 Detailed Description

Housekeeping Request command packet structure.

Definition at line 49 of file fm_msg.h.

11.161.2 Field Documentation

11.161.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_HousekeepingCmd_t::CmdHeader
```

Command header.

Definition at line 51 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.162 FM_HousekeepingPkt_t Struct Reference

Housekeeping telemetry packet.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_TelemetryHeader_t TlmHeader

  Telemetry Header.
- uint8 CommandCounter

  Application command counter.
- uint8 CommandErrCounter

  Application command error counter.
- uint8 Spare

  Placeholder for unused command warning counter.
- uint8 NumOpenFiles

  Number of open files in the system.
- uint8 ChildCmdCounter

  Child task command counter.
- uint8 ChildCmdErrCounter

  Child task command error counter.
- uint8 ChildCmdWarnCounter

  Child task command warning counter.
- uint8 ChildQueueCount

  Number of pending commands in queue.
- uint8 ChildCurrentCC

  Command code currently executing.
- uint8 ChildPreviousCC

  Command code previously executed.

11.162.1 Detailed Description

Housekeeping telemetry packet.

Definition at line 440 of file fm_msg.h.

11.162.2 Field Documentation

11.162.2.1 ChildCmdCounter

```
uint8 FM_HousekeepingPkt_t::ChildCmdCounter
```

Child task command counter.

Definition at line 450 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.2 ChildCmdErrCounter

```
uint8 FM_HousekeepingPkt_t::ChildCmdErrCounter
```

Child task command error counter.

Definition at line 451 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.3 ChildCmdWarnCounter

```
uint8 FM_HousekeepingPkt_t::ChildCmdWarnCounter
```

Child task command warning counter.

Definition at line 452 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.4 ChildCurrentCC

```
uint8 FM_HousekeepingPkt_t::ChildCurrentCC
```

Command code currently executing.

Definition at line 456 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.5 ChildPreviousCC

```
uint8 FM_HousekeepingPkt_t::ChildPreviousCC
```

Command code previously executed.

Definition at line 457 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.6 ChildQueueCount

```
uint8 FM_HousekeepingPkt_t::ChildQueueCount
```

Number of pending commands in queue.

Definition at line 454 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.7 CommandCounter

```
uint8 FM_HousekeepingPkt_t::CommandCounter
```

Application command counter.

Definition at line 444 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.8 CommandErrCounter

```
uint8 FM_HousekeepingPkt_t::CommandErrCounter
```

Application command error counter.

Definition at line 445 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.9 NumOpenFiles

```
uint8 FM_HousekeepingPkt_t::NumOpenFiles
```

Number of open files in the system.

Definition at line 448 of file fm_msg.h.

Referenced by FM_ReportHK().

11.162.2.10 Spare

```
uint8 FM_HousekeepingPkt_t::Spare
```

Placeholder for unused command warning counter.

Definition at line 446 of file fm_msg.h.

11.162.2.11 TlmHeader

```
CFE_MSG_TelemetryHeader_t FM_HousekeepingPkt_t::TlmHeader
```

Telemetry Header.

Definition at line 442 of file fm_msg.h.

Referenced by FM_ReportHK().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.163 FM_MoveFileCmd_t Struct Reference

Move File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- uint16 Overwrite

  Allow overwrite.
- char Source [OS_MAX_PATH_LEN]

  Source filename.
- char Target [OS_MAX_PATH_LEN]

  Target filename.

11.163.1 Detailed Description

Move File command packet structure.

For command details see FM_MOVE_CC

Definition at line 100 of file fm_msg.h.

11.163.2 Field Documentation

11.163.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_MoveFileCmd_t::CmdHeader
```

Command header.

Definition at line 102 of file fm_msg.h.

11.163.2.2 Overwrite

```
uint16 FM_MoveFileCmd_t::Overwrite
```

Allow overwrite.

Definition at line 104 of file fm_msg.h.

Referenced by FM_MoveFileCmd().

11.163.2.3 Source

```
char FM_MoveFileCmd_t::Source[OS_MAX_PATH_LEN]
```

Source filename.

Definition at line 105 of file fm_msg.h.

Referenced by FM_MoveFileCmd().

11.163.2.4 Target

```
char FM_MoveFileCmd_t::Target[OS_MAX_PATH_LEN]
```

Target filename.

Definition at line 106 of file fm_msg.h.

Referenced by FM_MoveFileCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.164 FM_NoopCmd_t Struct Reference

No-Operation command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.

11.164.1 Detailed Description

No-Operation command packet structure.

For command details see FM_NOOP_CC

Definition at line 60 of file fm_msg.h.

11.164.2 Field Documentation

11.164.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_NoopCmd_t::CmdHeader
```

Command header.

Definition at line 63 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.165 FM_OpenFilesEntry_t Struct Reference

Get Open Files list entry structure.

```
#include <fm_msg.h>
```

Data Fields

- char LogicalName [OS_MAX_PATH_LEN]

  Logical filename.
- char AppName [OS_MAX_API_NAME]

  Application that opened file.

11.165.1 Detailed Description

Get Open Files list entry structure.

Definition at line 388 of file fm_msg.h.

11.165.2 Field Documentation

11.165.2.1 AppName

```
char FM_OpenFilesEntry_t::AppName[OS_MAX_API_NAME]
```

Application that opened file.

Definition at line 391 of file fm_msg.h.

11.165.2.2 LogicalName

```
char FM_OpenFilesEntry_t::LogicalName[OS_MAX_PATH_LEN]
```

Logical filename.

Definition at line 390 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.166 FM_OpenFilesPkt_t Struct Reference

Get Open Files telemetry packet.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_TelemetryHeader_t TlmHeader

  Telemetry Header.
- uint32 NumOpenFiles

  Number of files opened via cFE.
- FM_OpenFilesEntry_t OpenFilesList [OS_MAX_NUM_OPEN_FILES]

  List of files opened via cFE.

11.166.1 Detailed Description

Get Open Files telemetry packet.

Definition at line 398 of file fm_msg.h.

11.166.2 Field Documentation

11.166.2.1 NumOpenFiles

```
uint32 FM_OpenFilesPkt_t::NumOpenFiles
```

Number of files opened via cFE.

Definition at line 402 of file fm_msg.h.

Referenced by FM_GetOpenFilesCmd().

11.166.2.2 OpenFilesList

```
FM_OpenFilesEntry_t FM_OpenFilesPkt_t::OpenFilesList[OS_MAX_NUM_OPEN_FILES]
```

List of files opened via cFE.

Definition at line 403 of file fm_msg.h.

Referenced by FM_GetOpenFilesCmd().

11.166.2.3 TlmHeader

`CFE_MSG_TelemetryHeader_t FM_OpenFilesPkt_t::TlmHeader`

Telemetry Header.

Definition at line 400 of file fm_msg.h.

Referenced by FM_GetOpenFilesCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.167 FM_RenameFileCmd_t Struct Reference

Rename File command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- char Source [OS_MAX_PATH_LEN]

  Source filename.
- char Target [OS_MAX_PATH_LEN]

  Target filename.

11.167.1 Detailed Description

Rename File command packet structure.

For command details see FM_RENAME_CC

Definition at line 115 of file fm_msg.h.

11.167.2 Field Documentation

11.167.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_RenameFileCmd_t::CmdHeader
```

Command header.

Definition at line 117 of file fm_msg.h.

11.167.2.2 Source

```
char FM_RenameFileCmd_t::Source[OS_MAX_PATH_LEN]
```

Source filename.

Definition at line 119 of file fm_msg.h.

Referenced by FM_RenameFileCmd().

11.167.2.3 Target

```
char FM_RenameFileCmd_t::Target[OS_MAX_PATH_LEN]
```

Target filename.

Definition at line 120 of file fm_msg.h.

Referenced by FM_RenameFileCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.168 FM_ResetCmd_t Struct Reference

Reset Counters command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.

11.168.1 Detailed Description

Reset Counters command packet structure.

For command details see FM_RESET_CC

Definition at line 72 of file fm_msg.h.

11.168.2 Field Documentation

11.168.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_ResetCmd_t::CmdHeader
```

Command header.

Definition at line 75 of file fm_msg.h.

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.169 FM_SetPermCmd_t Struct Reference

Set Permissions for a file.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader
  Command header.
- char FileName [OS_MAX_PATH_LEN]
  File name of the permissions to set.
- uint32 Mode
  Permissions, passed directly to OS_chmod.

11.169.1 Detailed Description

Set Permissions for a file.

For command details see FM_SET_FILE_PERM_CC

Definition at line 292 of file fm_msg.h.

11.169.2 Field Documentation

11.169.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_SetPermCmd_t::CmdHeader
```

Command header.

Definition at line 294 of file fm_msg.h.

11.169.2.2 FileName

```
char FM_SetPermCmd_t::FileName[OS_MAX_PATH_LEN]
```

File name of the permissions to set.

Definition at line 296 of file fm_msg.h.

Referenced by FM_SetPermissionsCmd().

11.169.2.3 Mode

```
uint32 FM_SetPermCmd_t::Mode
```

Permissions, passed directly to OS_chmod.

Definition at line 297 of file fm_msg.h.

Referenced by FM_SetPermissionsCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.170 FM_SetTableStateCmd_t Struct Reference

Set Table State command packet structure.

```
#include <fm_msg.h>
```

Data Fields

- CFE_MSG_CommandHeader_t CmdHeader

  Command header.
- uint32 TableEntryIndex

  Table entry index.
- uint32 TableEntryState

  New table entry state.

11.170.1 Detailed Description

Set Table State command packet structure.

For command details see FM_SET_TABLE_STATE_CC

Definition at line 278 of file fm_msg.h.

11.170.2 Field Documentation

11.170.2.1 CmdHeader

```
CFE_MSG_CommandHeader_t FM_SetTableStateCmd_t::CmdHeader
```

Command header.

Definition at line 280 of file fm_msg.h.

11.170.2.2 TableEntryIndex

```
uint32 FM_SetTableStateCmd_t::TableEntryIndex
```

Table entry index.

Definition at line 282 of file fm_msg.h.

Referenced by FM_SetTableStateCmd().

11.170.2.3 TableEntryState

```
uint32 FM_SetTableStateCmd_t::TableEntryState
```

New table entry state.

Definition at line 283 of file fm_msg.h.

Referenced by FM_SetTableStateCmd().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.171 FM_TableEntry_t Struct Reference

Get Free Space table entry.

```
#include <fm_msg.h>
```

Data Fields

- uint32 State

  Table entry enable/disable state.
- char Name [OS_MAX_PATH_LEN]

  File system name = string.

11.171.1 Detailed Description

Get Free Space table entry.

Definition at line 472 of file fm_msg.h.

11.171.2 Field Documentation

11.171.2.1 Name

```
char FM_TableEntry_t::Name[OS_MAX_PATH_LEN]
```

File system name = string.

Definition at line 475 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd(), and FM_ValidateTable().

11.171.2.2 State

```
uint32 FM_TableEntry_t::State
```

Table entry enable/disable state.

Definition at line 474 of file fm_msg.h.

Referenced by FM_GetFreeSpaceCmd(), FM_SetTableStateCmd(), and FM_ValidateTable().

The documentation for this struct was generated from the following file:

- apps/fm/fsw/src/fm_msg.h

11.172 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

11.172.1 Detailed Description

OSAL binary semaphore properties.

Definition at line 39 of file osapi-binsem.h.

11.172.2 Field Documentation

11.172.2.1 creator

```
osal_id_t OS_bin_sem_prop_t::creator
```

Definition at line 42 of file osapi-binsem.h.

11.172.2.2 name

```
char OS_bin_sem_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 41 of file osapi-binsem.h.

11.172.2.3 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

11.173 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

11.173.1 Detailed Description

OSAL counting semaphore properties.

Definition at line 32 of file osapi-countsem.h.

11.173.2 Field Documentation

11.173.2.1 creator

```
osal_id_t OS_count_sem_prop_t::creator
```

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

11.173.2.2 name

```
char OS_count_sem_prop_t::name[OS_MAX_API_NAME]
```

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

11.173.2.3 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

11.174 os_dirent_t Struct Reference

Directory entry.

```
#include <osapi-dir.h>
```

Data Fields

- char FileName [OS_MAX_FILE_NAME]

11.174.1 Detailed Description

Directory entry.

Definition at line 32 of file osapi-dir.h.

11.174.2 Field Documentation

11.174.2.1 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

11.175 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]

11.175.1 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.

11.175.2 Field Documentation

11.175.2.1 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

11.176 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

11.176.1 Detailed Description

OSAL file properties.

Definition at line 49 of file osapi-file.h.

11.176.2 Field Documentation

11.176.2.1 IsValid

```
uint8 OS_file_prop_t::IsValid
```

Definition at line 53 of file osapi-file.h.

11.176.2.2 Path

```
char OS_file_prop_t::Path[OS_MAX_PATH_LEN]
```

Definition at line 51 of file osapi-file.h.

Referenced by LoadOpenFileData(), and SearchOpenFileData().

11.176.2.3 User

```
osal_id_t OS_file_prop_t::User
```

Definition at line 52 of file osapi-file.h.

Referenced by LoadOpenFileData().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-file.h

11.177 os_fsinfo_t Struct Reference

OSAL file system info.

```
#include <osapi-filesys.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.

11.177.1 Detailed Description

OSAL file system info.

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

11.177.2 Field Documentation

11.177.2.1 FreeFds

```
uint32 os_fsinfo_t::FreeFds
```

Total number that are free.

Definition at line 38 of file osapi-filesys.h.

11.177.2.2 FreeVolumes

```
uint32 os_fsinfo_t::FreeVolumes
```

Total number of volumes free.

Definition at line 40 of file osapi-filesys.h.

11.177.2.3 MaxFds

```
uint32 os_fsinfo_t::MaxFds
```

Total number of file descriptors.

Definition at line 37 of file osapi-filesys.h.

11.177.2.4 MaxVolumes

```
uint32 os_fsinfo_t::MaxVolumes
```

Maximum number of volumes.

Definition at line 39 of file osapi-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-filesys.h

11.178 os_fstat_t Struct Reference

File system status.

```
#include <osapi-file.h>
```

Data Fields

- uint32 FileModeBits
- OS_time_t FileTime
- size_t FileSize

11.178.1 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.

11.178.2 Field Documentation

11.178.2.1 FileModeBits

```
uint32 os_fstat_t::FileModeBits
```

Definition at line 66 of file osapi-file.h.

11.178.2.2 FileSize

```
size_t os_fstat_t::FileSize
```

Definition at line 68 of file osapi-file.h.

11.178.2.3 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

11.179 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

11.179.1 Detailed Description

OSAL heap properties.

See also

OS_HeapGetInfo()

Definition at line 36 of file osapi-heap.h.

11.179.2 Field Documentation

11.179.2.1 free_blocks

```
osal_blockcount_t OS_heap_prop_t::free_blocks
```

Definition at line 39 of file osapi-heap.h.

11.179.2.2 free_bytes

```
size_t OS_heap_prop_t::free_bytes
```

Definition at line 38 of file osapi-heap.h.

11.179.2.3 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

11.180 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

11.180.1 Detailed Description

OSAL module address properties.

Definition at line 78 of file osapi-module.h.

11.180.2 Field Documentation

11.180.2.1 bss_address

```
cpuaddr OS_module_address_t::bss_address
```

Definition at line 86 of file osapi-module.h.

11.180.2.2 bss_size

```
cpuaddr OS_module_address_t::bss_size
```

Definition at line 87 of file osapi-module.h.

11.180.2.3 code_address

```
cpuaddr OS_module_address_t::code_address
```

Definition at line 82 of file osapi-module.h.

11.180.2.4 code_size

```
cpuaddr OS_module_address_t::code_size
```

Definition at line 83 of file osapi-module.h.

11.180.2.5 data_address

```
cpuaddr OS_module_address_t::data_address
```

Definition at line 84 of file osapi-module.h.

11.180.2.6 data_size

```
cpuaddr OS_module_address_t::data_size
```

Definition at line 85 of file osapi-module.h.

11.180.2.7 flags

```
uint32 OS_module_address_t::flags
```

Definition at line 81 of file osapi-module.h.

11.180.2.8 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

11.181 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

11.181.1 Detailed Description

OSAL module properties.

Definition at line 91 of file osapi-module.h.

11.181.2 Field Documentation

11.181.2.1 addr

```
OS_module_address_t OS_module_prop_t::addr
```

Definition at line 97 of file osapi-module.h.

11.181.2.2 entry_point

```
cpuaddr OS_module_prop_t::entry_point
```

Definition at line 93 of file osapi-module.h.

11.181.2.3 filename

```
char OS_module_prop_t::filename[OS_MAX_PATH_LEN]
```

Definition at line 95 of file osapi-module.h.

11.181.2.4 host_module_id

```
cpuaddr OS_module_prop_t::host_module_id
```

Definition at line 94 of file osapi-module.h.

11.181.2.5 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

11.182 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

11.182.1 Detailed Description

OSAL mutex properties.

Definition at line 32 of file osapi-mutex.h.

11.182.2 Field Documentation

11.182.2.1 creator

```
osal_id_t OS_mut_sem_prop_t::creator
```

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

11.182.2.2 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

11.183 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

11.183.1 Detailed Description

OSAL queue properties.

Definition at line 32 of file osapi-queue.h.

11.183.2 Field Documentation

11.183.2.1 creator

```
osal_id_t OS_queue_prop_t::creator
```

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

11.183.2.2 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

11.184 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.

11.184.1 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.

11.184.2 Field Documentation

11.184.2.1 ActualLength

```
size_t OS_SockAddr_t::ActualLength
```

Length of the actual address data.

Definition at line 111 of file osapi-sockets.h.

11.184.2.2 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

11.185 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.

11.185.1 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.

11.185.2 Field Documentation

11.185.2.1 AlignPtr

```
void* OS_SockAddrData_t::AlignPtr
```

Ensures pointer alignment.

Definition at line 99 of file osapi-sockets.h.

11.185.2.2 AlignU32

```
uint32 OS_SockAddrData_t::AlignU32
```

Ensures uint32 alignment.

Definition at line 98 of file osapi-sockets.h.

11.185.2.3 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

11.186 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.

11.186.1 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.

11.186.2 Field Documentation

11.186.2.1 creator

```
osal_id_t OS_socket_prop_t::creator
```

OSAL TaskID which opened the socket.

Definition at line 125 of file osapi-sockets.h.

11.186.2.2 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

11.187 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

11.187.1 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.

11.187.2 Field Documentation

11.187.2.1 Address

```
void(* OS_static_symbol_record_t::Address) (void)
```

Definition at line 116 of file osapi-module.h.

11.187.2.2 Module

```
const char* OS_static_symbol_record_t::Module
```

Definition at line 117 of file osapi-module.h.

11.187.2.3 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

11.188 OS_statvfs_t Struct Reference

```
#include <osapi-filesys.h>
```

Data Fields

- size_t block_size
- osal_blockcount_t total_blocks
- osal_blockcount_t blocks_free

11.188.1 Detailed Description

Definition at line 49 of file osapi-filesys.h.

11.188.2 Field Documentation

11.188.2.1 block_size

```
size_t OS_statvfs_t::block_size
```

Block size of underlying FS

Definition at line 51 of file osapi-filesys.h.

11.188.2.2 blocks_free

```
osal_blockcount_t OS_statvfs_t::blocks_free
```

Available blocks in underlying FS

Definition at line 53 of file osapi-filesys.h.

Referenced by FM_GetFreeSpaceCmd().

11.188.2.3 total_blocks

```
osal_blockcount_t OS_statvfs_t::total_blocks
```

Total blocks in underlying FS

Definition at line 52 of file osapi-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-filesys.h

11.189 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

11.189.1 Detailed Description

OSAL task properties.

Definition at line 57 of file osapi-task.h.

11.189.2 Field Documentation

11.189.2.1 creator

```
osal_id_t OS_task_prop_t::creator
```

Definition at line 60 of file osapi-task.h.

11.189.2.2 name

```
char OS_task_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 59 of file osapi-task.h.

Referenced by LoadOpenFileData().

11.189.2.3 priority

```
osal_priority_t OS_task_prop_t::priority
```

Definition at line 62 of file osapi-task.h.

11.189.2.4 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

11.190 OS_time_t Struct Reference

OSAL time interval structure.

```
#include <osapi-clock.h>
```

Data Fields

- int64 ticks

11.190.1 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.

11.190.2 Field Documentation

11.190.2.1 ticks

```
int64 OS_time_t::ticks
```

Ticks elapsed since reference point

Definition at line 47 of file osapi-clock.h.

Referenced by OS_TimeAdd(), OS_TimeAssembleFromMicroseconds(), OS_TimeAssembleFromMilliseconds(), OS_TimeAssembleFromNanoseconds(), OS_TimeAssembleFromSubseconds(), OS_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

11.191 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

11.191.1 Detailed Description

Time base properties.

Definition at line 37 of file osapi-timebase.h.

11.191.2 Field Documentation

11.191.2.1 accuracy

```
uint32 OS_timebase_prop_t::accuracy
```

Definition at line 43 of file osapi-timebase.h.

11.191.2.2 creator

```
osal_id_t OS_timebase_prop_t::creator
```

Definition at line 40 of file osapi-timebase.h.

11.191.2.3 freerun_time

```
uint32 OS_timebase_prop_t::freerun_time
```

Definition at line 42 of file osapi-timebase.h.

11.191.2.4 name

```
char OS_timebase_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 39 of file osapi-timebase.h.

11.191.2.5 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

11.192 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

11.192.1 Detailed Description

Timer properties.

Definition at line 37 of file osapi-timer.h.

11.192.2 Field Documentation

11.192.2.1 accuracy

```
uint32 OS_timer_prop_t::accuracy
```

Definition at line 43 of file osapi-timer.h.

11.192.2.2 creator

```
osal_id_t OS_timer_prop_t::creator
```

Definition at line 40 of file osapi-timer.h.

11.192.2.3 interval_time

```
uint32 OS_timer_prop_t::interval_time
```

Definition at line 42 of file osapi-timer.h.

11.192.2.4 name

```
char OS_timer_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 39 of file osapi-timer.h.

11.192.2.5 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/fm/docs/dox_src/cfs_fm.dox File Reference

12.2 apps/fm/fsw/mission_inc/fm_perfids.h File Reference

Macros

- #define FM_APPMAIN_PERF_ID 39

  Main application performance ID.
- #define FM_CHILD_TASK_PERF_ID 44

  Child task performance ID.

12.2.1 Detailed Description

Specification for the CFS File Manager (FM) Application Performance IDs

12.3 apps/fm/fsw/platform_inc/fm_msgids.h File Reference

Macros

- #define FM_CMD_MID 0x188C /** < \brief FM ground commands */
- #define FM_SEND_HK_MID 0x188D /** < \brief FM send housekeeping */
- #define FM_HK_TLM_MID 0x088A /** < \brief FM housekeeping */
- #define FM_FILE_INFO_TLM_MID 0x088B /** < \brief FM get file info */
- #define FM_DIR_LIST_TLM_MID 0x088C /** < \brief FM get dir list */
- #define FM_OPEN_FILES_TLM_MID 0x088D /** < \brief FM get open files */
- #define FM_FREE_SPACE_TLM_MID 0x088E /** < \brief FM get free space */

12.3.1 Detailed Description

Specification for the CFS FM application software bus message identifiers

12.4 apps/fm/fsw/platform_inc/fm_platform_cfg.h File Reference

Macros

- #define FM_APP_NAME "FM"

  File Manager Application Name.
- #define FM_APP_PIPE_NAME "FM_CMD_PIPE"

  File Manager Command Pipe Name.
- #define FM_APP_PIPE_DEPTH 10

  File Manager Command Pipe Depth.
- #define FM_MISSION_REV 0

  Mission specific version number for FM application.
- #define FM_DIR_LIST_FILE_DEFNAME "/ram/fm_dirlist.out"

  Default Directory List Output Filename.
- #define FM_DIR_LIST_FILE_ENTRIES 3000

  Maximum Directory List Output File Entries.
- #define FM_DIR_LIST_FILE_SUBTYPE 12345

  Directory List Output File Header Sub-Type.
- #define FM_DIR_LIST_PKT_ENTRIES 20

  Directory List Telemetry Packet Entry Count.
- #define FM_CHILD_FILE_BLOCK_SIZE 2048

  Child Task File I/O Control Settings.
- #define FM_CHILD_FILE_LOOP_COUNT 16
- #define FM_CHILD_FILE_SLEEP_MS 20
- #define FM_CHILD_STAT_SLEEP_MS 0

  Child file stat sleep.
- #define FM_CHILD_STAT_SLEEP_FILECOUNT 0
- #define FM_CHILD_QUEUE_DEPTH 3

  Child Task Command Queue Entry Count.
- #define FM_CHILD_TASK_NAME "FM_CHILD_TASK"

  Child Task Name - cFE object name.
- #define FM_CHILD_TASK_STACK_SIZE 20480

  Child Task Stack Size.
- #define FM_CHILD_TASK_PRIORITY 205

  Child Task Execution Priority.
- #define FM_CHILD_SEM_NAME "FM_CHILD_SEM"

  Child Task Semaphore Name - cFE object name.
- #define FM_TABLE_CFE_NAME "FreeSpace"

  Free Space Table Name - cFE object name.
- #define FM_TABLE_DEF_NAME "/cf/fm_freespace.tbl"

  Free Space Table Name - filename with path.
- #define FM_TABLE_FILENAME "fm_freespace.tbl"

Free Space Table Name - filename without path.

- #define FM_TABLE_DEF_DESC "FM File System Free Space Table"

  Free Space Table Description.

- #define FM_TABLE_ENTRY_COUNT 8

  Number of Free Space Table Entries.

- #define FM_TABLE_VALIDATION_ERR (-1)

  Table Data Validation Error Code.

12.4.1 Detailed Description

Specification for the CFS FM application constants that can be configured from one platform to another

12.5 apps/fm/fsw/src/fm_app.c File Reference

```
#include "cfe.h"
#include "fm_msg.h"
#include "fm_msgdefs.h"
#include "fm_msgids.h"
#include "fm_app.h"
#include "fm_tbl.h"
#include "fm_child.h"
#include "fm_cmds.h"
#include "fm_cmd_utils.h"
#include "fm_events.h"
#include "fm_perfids.h"
#include "fm_platform_cfg.h"
#include "fm_version.h"
#include "fm_verify.h"
#include <string.h>
```

Functions

- void FM_AppMain (void)

  Application entry point and main process loop.

- int32 FM_AppInit (void)

  FM Application Initialization Function.

- void FM_ProcessPkt (const CFE_SB_Buffer_t *BufPtr)

  Process Input Command Packets.

- void FM_ProcessCmd (const CFE_SB_Buffer_t *BufPtr)

  Process FM Ground Commands.

- void FM_ReportHK (const CFE_MSG_CommandHeader_t *Msg)

  Housekeeping Request Command Handler.

Variables

- FM_GlobalData_t FM_GlobalData

  File Manager global.

12.5.1 Detailed Description

Core Flight System (CFS) File Manager (FM) Application

The File Manager (FM) Application provides onboard file system management services by processing commands for copying and moving files, decompressing files, concatenating files, creating directories, deleting files and directories, and providing file and directory status. When the File Manager application receives a housekeeping request (scheduled within the scheduler application), FM reports it's housekeeping status values via telemetry messaging.

12.5.2 Function Documentation

12.5.2.1 FM_AppInit()

```
int32 FM_AppInit (
            void  )
```

FM Application Initialization Function.

Description

```
Initialize FM global data structure.
Register FM application for CFE Event Services.
Create Software Bus input pipe.
Subscribe to FM housekeeping request command packet.
Subscribe to FM ground command packet.
Invoke FM table initialization function.
Invoke FM child task initialization function.
```

Assumptions, External Events, and Notes: None

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

CFE_EVS_Register, CFE_SB_CreatePipe, CFE_SB_Subscribe

Definition at line 168 of file fm_app.c.

References CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_PSP_MemSet(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, FM_GlobalData_t::CmdPipe, FM_APP_PIPE_DEPTH, FM_APP_PIPE↩_NAME, FM_ChildInit(), FM_CMD_MID, FM_MAJOR_VERSION, FM_MINOR_VERSION, FM_MISSION_REV, FM_↩REVISION, FM_SEND_HK_MID, FM_STARTUP_CREAT_PIPE_ERR_EID, FM_STARTUP_EID, FM_STARTUP_E↩VENTS_ERR_EID, FM_STARTUP_SUBSCRIB_GCMD_ERR_EID, FM_STARTUP_SUBSCRIB_HK_ERR_EID, FM↩_STARTUP_TABLE_INIT_ERR_EID, and FM_TableInit().

Referenced by FM_AppMain().

Here is the call graph for this function:

12.5.2.2 FM_AppMain()

```
void FM_AppMain (
            void  )
```

Application entry point and main process loop.

Description

```
Register FM as a CFE application.
Invoke FM application initialization function.
Enter FM main process loop.
  Pend (forever) on next Software Bus command packet.
  Process received Software Bus command packet.
  Repeat main process loop.
Allow CFE to terminate the FM application.
```

Assumptions, External Events, and Notes: None

See also

FM_AppInit, CFE_ES_RunLoop, FM_ProcessPkt

Definition at line 64 of file fm_app.c.

References CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_Run↩Status_APP_ERROR, CFE_ES_RunStatus_APP_RUN, CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_ReceiveBuffer(), CFE_SB_TIME_OUT, CFE_SUCCESS, FM_GlobalData_t::Cmd↩Pipe, FM_AcquireTablePointers(), FM_AppInit(), FM_APPMAIN_PERF_ID, FM_EXIT_ERR_EID, FM_ProcessPkt(), F↩M_ReleaseTablePointers(), FM_SB_RECEIVE_ERR_EID, FM_SB_RECEIVE_NULL_PTR_ERR_EID, and FM_SB_T↩IMEOUT.

Here is the call graph for this function:

12.5.2.3 FM_ProcessCmd()

```
void FM_ProcessCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Process FM Ground Commands.

Description

```
Branch to the command specific handlers for FM ground commands.
```

Assumptions, External Events, and Notes: None

Parameters

in	*BufPtr*	Pointer to Software Bus message buffer.

Definition at line 284 of file fm_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), FM_GlobalData_↩t::CommandCounter, FM_GlobalData_t::CommandErrCounter, FM_CC_ERR_EID, FM_CONCAT_CC, FM_Concat↩FilesCmd(), FM_COPY_CC, FM_CopyFileCmd(), FM_CREATE_DIR_CC, FM_CreateDirectoryCmd(), FM_DECOMP↩RESS_CC, FM_DELETE_ALL_CC, FM_DELETE_CC, FM_DELETE_DIR_CC, FM_DELETE_INT_CC, FM_DeleteAll↩FilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC, F↩M_GET_FILE_INFO_CC, FM_GET_FREE_SPACE_CC, FM_GET_OPEN_FILES_CC, FM_GetDirListFileCmd(), F↩M_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_MOVE_CC, FM_MoveFileCmd(), FM_NOOP_CC, FM_NoopCmd(), FM_RENAME_CC, FM_RenameFileCmd(), FM_RESET_CC, FM_ResetCountersCmd(), FM_SET_FILE_PERM_CC, FM_SET_TABLE_STATE_CC, FM_SetPermissionsCmd(), F↩M_SetTableStateCmd(), and CFE_SB_Msg::Msg.

Referenced by FM_ProcessPkt().

Here is the call graph for this function:

12.5.2.4 FM_ProcessPkt()

```
void FM_ProcessPkt (
            const CFE_SB_Buffer_t * MessagePtr )
```

Process Input Command Packets.

Description

```
Branch to appropriate input packet handler: HK request or FM commands.
```

Assumptions, External Events, and Notes: None

Parameters

in	*MessagePtr*	Pointer to Software Bus message buffer.

See also

FM_ReportHK, FM_ProcessCmd

Definition at line 250 of file fm_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetMsgId(), CFE_SB_INVALID_↩MSG_ID, CFE_SB_MsgIdToValue(), FM_CMD_MID, FM_MID_ERR_EID, FM_ProcessCmd(), FM_ReportHK(), FM_↩SEND_HK_MID, and CFE_SB_Msg::Msg.

Referenced by FM_AppMain().

Here is the call graph for this function:

12.5.2.5 FM_ReportHK()

```
void FM_ReportHK (
            const CFE_MSG_CommandHeader_t * Msg )
```

Housekeeping Request Command Handler.

Description

```
Allow CFE Table Services the opportunity to manage the File System
Free Space Table.  This provides a mechanism to receive table updates.

Populate the FM application Housekeeping Telemetry packet.  Timestamp
the packet and send it to ground via the Software Bus.
```

Assumptions, External Events, and Notes: None

Parameters

in	*Msg*	Pointer to Software Bus command packet.

See also

FM_HousekeepingCmd_t, FM_HousekeepingPkt_t

Definition at line 400 of file fm_app.c.

References CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_HousekeepingPkt_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdCounter, FM_HousekeepingPkt_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_HousekeepingPkt_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_HousekeepingPkt_t::ChildCurrentCC, FM_GlobalData_t::ChildCurrentCC, FM_HousekeepingPkt_t::ChildPreviousCC, FM_GlobalData_t::ChildPreviousCC, FM_HousekeepingPkt_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCount, FM_HousekeepingPkt_t::CommandCounter, FM_GlobalData_t::CommandCounter, FM_HousekeepingPkt_t::CommandErrCounter, FM_GlobalData_t::CommandErrCounter, FM_AcquireTablePointers(), FM_GetOpenFilesData(), FM_HK_REQ_ERR_EID, FM_HK_TLM_MID, FM_IsValidCmdPktLength(), FM_ReleaseTablePointers(), FM_GlobalData_t::HousekeepingPkt, FM_HousekeepingPkt_t::NumOpenFiles, and FM_HousekeepingPkt_t::TlmHeader.

Referenced by FM_ProcessPkt().

Here is the call graph for this function:

12.5.3 Variable Documentation

12.5.3.1 FM_GlobalData

```
FM_GlobalData_t FM_GlobalData
```

File Manager global.

Definition at line 56 of file fm_app.c.

Referenced by FM_AcquireTablePointers(), FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_Child↩DirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildInit(), F↩M_ChildLoop(), FM_ChildMoveCmd(), FM_ChildProcess(), FM_ChildRenameCmd(), FM_ChildSetPermissions↩Cmd(), FM_ChildTask(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAll↩FilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFilenameState(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_Invoke↩ChildTask(), FM_MoveFileCmd(), FM_ReleaseTablePointers(), FM_RenameFileCmd(), FM_ResetCountersCmd(), FM_SetPermissionsCmd(), FM_SetTableStateCmd(), FM_TableInit(), and FM_VerifyChildTask().

12.6 apps/fm/fsw/src/fm_app.h File Reference

```
#include "cfe.h"
```

Macros

- #define FM_SB_TIMEOUT 1000

  Wakeup for FM.

Functions

- void FM_AppMain (void)

  Application entry point and main process loop.
- int32 FM_AppInit (void)

  FM Application Initialization Function.
- void FM_ProcessPkt (const CFE_SB_Buffer_t *MessagePtr)

  Process Input Command Packets.
- void FM_ProcessCmd (const CFE_SB_Buffer_t *BufPtr)

  Process FM Ground Commands.
- void FM_ReportHK (const CFE_MSG_CommandHeader_t *Msg)

  Housekeeping Request Command Handler.

12.6.1 Detailed Description

Unit specification for the CFS File Manager Application.

12.6.2 Macro Definition Documentation

12.6.2.1 FM_SB_TIMEOUT

```
#define FM_SB_TIMEOUT 1000
```

Wakeup for FM.

Description

Wakes up FM every 1 second for routine maintenance whether a message was received or not.

Definition at line 42 of file fm_app.h.

Referenced by FM_AppMain().

12.6.3 Function Documentation

12.6.3.1 FM_AppInit()

```
int32 FM_AppInit (
            void  )
```

FM Application Initialization Function.

Description

```
Initialize FM global data structure.
Register FM application for CFE Event Services.
Create Software Bus input pipe.
Subscribe to FM housekeeping request command packet.
Subscribe to FM ground command packet.
Invoke FM table initialization function.
Invoke FM child task initialization function.
```

Assumptions, External Events, and Notes: None

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

CFE_EVS_Register, CFE_SB_CreatePipe, CFE_SB_Subscribe

Definition at line 168 of file fm_app.c.

References CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_PSP_MemSet(), CFE_SB_CreatePipe(), CFE_SB_Subscribe(), CFE_SB_ValueToMsgId(), CFE_SUCCESS, FM_GlobalData_t::CmdPipe, FM_APP_PIPE_DEPTH, FM_APP_PIPE↩_NAME, FM_ChildInit(), FM_CMD_MID, FM_MAJOR_VERSION, FM_MINOR_VERSION, FM_MISSION_REV, FM_↩REVISION, FM_SEND_HK_MID, FM_STARTUP_CREAT_PIPE_ERR_EID, FM_STARTUP_EID, FM_STARTUP_E↩VENTS_ERR_EID, FM_STARTUP_SUBSCRIB_GCMD_ERR_EID, FM_STARTUP_SUBSCRIB_HK_ERR_EID, FM↩_STARTUP_TABLE_INIT_ERR_EID, and FM_TableInit().

Referenced by FM_AppMain().

Here is the call graph for this function:

12.6.3.2 FM_AppMain()

```
void FM_AppMain (
            void  )
```

Application entry point and main process loop.

Description

```
Register FM as a CFE application.
Invoke FM application initialization function.
Enter FM main process loop.
  Pend (forever) on next Software Bus command packet.
  Process received Software Bus command packet.
  Repeat main process loop.
Allow CFE to terminate the FM application.
```

Assumptions, External Events, and Notes: None

See also

FM_AppInit, CFE_ES_RunLoop, FM_ProcessPkt

Definition at line 64 of file fm_app.c.

References CFE_ES_ExitApp(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunLoop(), CFE_ES_Run↩Status_APP_ERROR, CFE_ES_RunStatus_APP_RUN, CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_ReceiveBuffer(), CFE_SB_TIME_OUT, CFE_SUCCESS, FM_GlobalData_t::Cmd↩Pipe, FM_AcquireTablePointers(), FM_AppInit(), FM_APPMAIN_PERF_ID, FM_EXIT_ERR_EID, FM_ProcessPkt(), F↩M_ReleaseTablePointers(), FM_SB_RECEIVE_ERR_EID, FM_SB_RECEIVE_NULL_PTR_ERR_EID, and FM_SB_T↩IMEOUT.

Here is the call graph for this function:

12.6.3.3 FM_ProcessCmd()

```
void FM_ProcessCmd (
```

```
const CFE_SB_Buffer_t * BufPtr )
```

Process FM Ground Commands.

Description

```
Branch to the command specific handlers for FM ground commands.
```

Assumptions, External Events, and Notes: None

Parameters

in	*BufPtr*	Pointer to Software Bus message buffer.

Definition at line 284 of file fm_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetFcnCode(), FM_GlobalData_↩t::CommandCounter, FM_GlobalData_t::CommandErrCounter, FM_CC_ERR_EID, FM_CONCAT_CC, FM_Concat↩FilesCmd(), FM_COPY_CC, FM_CopyFileCmd(), FM_CREATE_DIR_CC, FM_CreateDirectoryCmd(), FM_DECOMP↩RESS_CC, FM_DELETE_ALL_CC, FM_DELETE_CC, FM_DELETE_DIR_CC, FM_DELETE_INT_CC, FM_DeleteAll↩FilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC, F↩M_GET_FILE_INFO_CC, FM_GET_FREE_SPACE_CC, FM_GET_OPEN_FILES_CC, FM_GetDirListFileCmd(), F↩M_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_MOVE_CC, FM_MoveFileCmd(), FM_NOOP_CC, FM_NoopCmd(), FM_RENAME_CC, FM_RenameFileCmd(), FM_RESET_CC, FM_ResetCountersCmd(), FM_SET_FILE_PERM_CC, FM_SET_TABLE_STATE_CC, FM_SetPermissionsCmd(), F↩M_SetTableStateCmd(), and CFE_SB_Msg::Msg.

Referenced by FM_ProcessPkt().

Here is the call graph for this function:

12.6.3.4 FM_ProcessPkt()

```
void FM_ProcessPkt (
            const CFE_SB_Buffer_t * MessagePtr )
```

Process Input Command Packets.

Description

```
Branch to appropriate input packet handler: HK request or FM commands.
```

Assumptions, External Events, and Notes: None

Parameters

in	*MessagePtr*	Pointer to Software Bus message buffer.

See also

FM_ReportHK, FM_ProcessCmd

Definition at line 250 of file fm_app.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_GetMsgId(), CFE_SB_INVALID_↩MSG_ID, CFE_SB_MsgIdToValue(), FM_CMD_MID, FM_MID_ERR_EID, FM_ProcessCmd(), FM_ReportHK(), FM_↩SEND_HK_MID, and CFE_SB_Msg::Msg.

Referenced by FM_AppMain().

Here is the call graph for this function:

12.6.3.5 FM_ReportHK()

```
void FM_ReportHK (
            const CFE_MSG_CommandHeader_t * Msg )
```

Housekeeping Request Command Handler.

Description

```
Allow CFE Table Services the opportunity to manage the File System
Free Space Table.  This provides a mechanism to receive table updates.

Populate the FM application Housekeeping Telemetry packet.  Timestamp
the packet and send it to ground via the Software Bus.
```

Assumptions, External Events, and Notes: None

Parameters

in	*Msg*	Pointer to Software Bus command packet.

See also

FM_HousekeepingCmd_t, FM_HousekeepingPkt_t

Definition at line 400 of file fm_app.c.

References CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_HousekeepingPkt_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdCounter, FM_HousekeepingPkt_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_HousekeepingPkt_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_HousekeepingPkt_t::ChildCurrentCC, FM_GlobalData_t::ChildCurrentCC, FM_HousekeepingPkt_t::ChildPreviousCC, FM_GlobalData_t::ChildPreviousCC, FM_HousekeepingPkt_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCount, FM_HousekeepingPkt_t::CommandCounter, FM_GlobalData_t::CommandCounter, FM_HousekeepingPkt_t::CommandErrCounter, FM_GlobalData_t::CommandErrCounter, FM_AcquireTablePointers(), FM_GetOpenFilesData(), FM_HK_REQ_ERR_EID, FM_HK_TLM_MID, FM_IsValidCmdPktLength(), FM_ReleaseTablePointers(), FM_GlobalData_t::HousekeepingPkt, FM_HousekeepingPkt_t::NumOpenFiles, and FM_HousekeepingPkt_t::TlmHeader.

Referenced by FM_ProcessPkt().

Here is the call graph for this function:

12.7 apps/fm/fsw/src/fm_child.c File Reference

```
#include "cfe.h"
#include "fm_msg.h"
#include "fm_msgdefs.h"
#include "fm_msgids.h"
#include "fm_events.h"
#include "fm_app.h"
#include "fm_child.h"
#include "fm_cmds.h"
#include "fm_cmd_utils.h"
#include "fm_perfids.h"
#include "fm_platform_cfg.h"
#include "fm_verify.h"
#include <string.h>
```

Macros

- #define OS_DIRENTRY_NAME(x) ((x).d_name)
- #define FM_QUEUE_SEM_NAME "FM_QUEUE_SEM"

Functions

- int32 FM_ChildInit (void)

  Child Task Initialization Function.
- void FM_ChildTask (void)

  Child Task Entry Point Function.
- void FM_ChildLoop (void)

  Child Task Main Loop Processor Function.
- void FM_ChildProcess (void)

  Child Task Command Queue Processor Function.
- void FM_ChildCopyCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Copy File Command Handler.
- void FM_ChildMoveCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Move File Command Handler.
- void FM_ChildRenameCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Rename File Command Handler.
- void FM_ChildDeleteCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete File Command Handler.
- void FM_ChildDeleteAllCmd (FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete All Files Command Handler.
- void FM_ChildConcatCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Concatenate Files Command Handler.
- void FM_ChildFileInfoCmd (FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get File Info Command Handler.
- void FM_ChildCreateDirCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Create Directory Command Handler.

- void FM_ChildDeleteDirCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete Directory Command Handler.
- void FM_ChildDirListFileCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get Dir List to File Command Handler.
- void FM_ChildDirListPktCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get Dir List to Packet Command Handler.
- void FM_ChildSetPermissionsCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Set Permissions Command Handler.
- bool FM_ChildDirListFileInit (osal_id_t *FileHandlePtr, const char *Directory, const char *Filename)

  Child Task Get Dir List to File Initialization Function.
- void FM_ChildDirListFileLoop (osal_id_t DirId, osal_id_t FileHandle, const char *Directory, const char *DirWith↩Sep, const char *Filename, uint8 getSizeTimeMode)

  Child Task Get Dir List to File Loop Processor Function.
- int32 FM_ChildSizeTimeMode (const char *Filename, uint32 *FileSize, uint32 *FileTime, uint32 *FileMode)

  Child Task File Size Time and Mode Utility Function.
- void FM_ChildSleepStat (const char *Filename, FM_DirListEntry_t *DirListData, int32 *FilesTillSleep, bool get↩SizeTimeMode)

  Child Task Sleep and Stat Utility Function.

12.7.1 Detailed Description

File Manager (FM) Child task (low priority command handler)

12.7.2 Macro Definition Documentation

12.7.2.1 FM_QUEUE_SEM_NAME

```
#define FM_QUEUE_SEM_NAME "FM_QUEUE_SEM"
```

Definition at line 53 of file fm_child.c.

Referenced by FM_ChildInit().

12.7.2.2 OS_DIRENTRY_NAME

```
#define OS_DIRENTRY_NAME(
            x ) ((x).d_name)
```

Definition at line 50 of file fm_child.c.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileLoop(), and FM_ChildDirList↩PktCmd().

12.7.3 Function Documentation

12.7.3.1 FM_ChildConcatCmd()

```
void FM_ChildConcatCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Concatenate Files Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a concatenate files command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_ConcatCmd_t

Definition at line 670 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildBuffer, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_CHILD_FILE_BLOCK_SIZE, FM_CHILD_FILE_LOOP_COUNT, FM_CHILD_FILE_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_CONCAT_CMD_EID, FM_CONCAT_OPEN_SRC2_ERR_EID, FM_CONCAT_OPEN_TGT_ERR_EID, FM_CONCAT_OSCPY_ERR_EID, FM_CONCAT_OSRD_ERR_EID, FM_CONCAT_OSWR_ERR_EID, FM_GlobalData, OS_close(), OS_cp(), OS_FILE_FLAG_NONE, OS_lseek(), OS_OBJECT_ID_UNDEFINED, OS_OpenCreate(), OS_read(), OS_READ_ONLY, OS_READ_WRITE, OS_remove(), OS_SEEK_END, OS_SUCCESS, OS_TaskDelay(), OS_write(), FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.2 FM_ChildCopyCmd()

```
void FM_ChildCopyCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Copy File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a copy file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_CopyFileCmd_t

Definition at line 299 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_COPY_CMD_EID, FM_COPY_OS_ERR_EID, FM_GlobalData, OS_cp(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.3 FM_ChildCreateDirCmd()

```
void FM_ChildCreateDirCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Create Directory Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a create directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_CreateDirCmd_t

Definition at line 1008 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_CREATE_DIR_CMD_EID, FM_CREATE_D↩IR_OS_ERR_EID, FM_GlobalData, OS_mkdir(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.4 FM_ChildDeleteAllCmd()

```
void FM_ChildDeleteAllCmd (
            FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete All Files Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete all files from a directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteAllCmd_t

Definition at line 471 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DECOM_CFE_ERR_EID, FM_DECOM_CMD_EID, FM_DELETE_ALL_CMD_EID, FM_DELETE_ALL_FILES_ND_WARNING_EID, FM_DELETE_ALL_OS_ERR_EID, FM_DELETE_ALL_SKIP_WARNING_EID, FM_GetFilenameState(), FM_GlobalData, FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, FM_NAME_IS_NOT_IN_USE, FM_PARENT_DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DirectoryRewind(), OS_DIRENTRY_NAME, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_remove(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.5 FM_ChildDeleteCmd()

```
void FM_ChildDeleteCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteFileCmd_t

Definition at line 426 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DELETE_CMD_EID, FM_DELETE_INT_CC, FM_DELETE_OS_ERR_EID, FM_GlobalData, OS_remove(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.6 FM_ChildDeleteDirCmd()

```
void FM_ChildDeleteDirCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete Directory Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteDirCmd_t

Definition at line 1050 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DELETE_DIR_CMD_EID, FM_DELETE_DIR_EMPTY_ERR_EID, FM_DELETE_OPENDIR_OS_ERR_EID, FM_DELETE_RMDIR_OS_ERR_EID, FM_GlobalData, FM_PARENT_DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_OBJECT_ID_UNDEFINED, OS_rmdir(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.7 FM_ChildDirListFileCmd()

```
void FM_ChildDirListFileCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get Dir List to File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get directory listing to a file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetDirFileCmd_t

Definition at line 1129 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_ChildDirListFileInit(), FM_ChildDirListFileLoop(), FM_GET_DIR_FILE_OSOPENDIR_ERR_EID, FM_GlobalData, FM_ChildQueueEntry_t::GetSizeTimeMode, OS_close(), OS_DirectoryClose(), OS_DirectoryOpen(), OS_OBJECT_ID_UNDEFINED, OS_SUCCESS, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.8 FM_ChildDirListFileInit()

```
bool FM_ChildDirListFileInit (
            osal_id_t * FileHandlePtr,
            const char * Directory,
            const char * Filename )
```

Child Task Get Dir List to File Initialization Function.

Description

This function creates the output file and then writes both the CFE file header and a blank copy of the directory list statistics structure to the output file. At the end of the command, software will re-write the statistics structure, this time with up to date values.

Assumptions, External Events, and Notes:

Parameters

out	*FileHandlePtr*	A pointer to a file handle variable which is modified to contain the newly created output file handle.
in	*Directory*	A pointer to a buffer containing the directory name.
in	*Filename*	A pointer to a buffer containing the output filename.

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1363 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_WriteHeader(), CFE_PSP_MemSet(), FM_GlobalData_t::ChildCmdErrCounter, CFE_FS_Header::Description, FM_GlobalData_t::DirListFileStats, FM_Dir↩ListFileStats_t::DirName, FM_DIR_LIST_FILE_SUBTYPE, FM_GET_DIR_FILE_OSCREAT_ERR_EID, FM_GET_D↩IR_FILE_WRBLANK_ERR_EID, FM_GET_DIR_FILE_WRHDR_ERR_EID, FM_GlobalData, OS_close(), OS_FILE_↩FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_Open↩Create(), OS_READ_WRITE, OS_SUCCESS, OS_write(), and CFE_FS_Header::SubType.

Referenced by FM_ChildDirListFileCmd().

Here is the call graph for this function:

12.7.3.9 FM_ChildDirListFileLoop()

```
void FM_ChildDirListFileLoop (
            osal_id_t DirId,
            osal_id_t FileHandle,
            const char * Directory,
            const char * DirWithSep,
            const char * Filename,
            uint8 GetSizeTimeMode )
```

Child Task Get Dir List to File Loop Processor Function.

Description

This function reads each directory entry, determines the last modify time size and mode for each entry, and writes the entry data to the output file.

Assumptions, External Events, and Notes:

Parameters

in	*DirId*	Directory ID, a handle used to read directory entries.
in	*FileHandle*	Output file handle.
in	*Directory*	Pointer to a buffer containing the directory name.
in	*DirWithSep*	Pointer to directory name with path separator appended.
in	*Filename*	Pointer to a buffer containing the output filename.
in	*GetSizeTimeMode*	Option to call OS_stat for size, time, mode of files

Definition at line 1447 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_Global↩Data_t::ChildCmdWarnCounter, FM_DirListFileStats_t::DirEntries, FM_GlobalData_t::DirListFileStats, FM_DirList↩Entry_t::EntryName, FM_DirListFileStats_t::FileEntries, FM_CHILD_STAT_SLEEP_FILECOUNT, FM_ChildSleep↩Stat(), FM_DIR_LIST_FILE_ENTRIES, FM_GET_DIR_FILE_CMD_EID, FM_GET_DIR_FILE_UPSTATS_ERR_EID, FM_GET_DIR_FILE_WARNING_EID, FM_GET_DIR_FILE_WRENTRY_ERR_EID, FM_GlobalData, FM_PARENT_↩DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_lseek(), OS_MAX_PATH↩_LEN, OS_SEEK_SET, OS_SUCCESS, and OS_write().

Referenced by FM_ChildDirListFileCmd().

Here is the call graph for this function:

12.7.3.10 FM_ChildDirListPktCmd()

```
void FM_ChildDirListPktCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get Dir List to Packet Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get directory listing to a packet command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetDirPktCmd_t

Definition at line 1192 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_Value↩ ToMsgId(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::Child↩ CmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry↩ _t::CommandCode, FM_ChildQueueEntry_t::DirListOffset, FM_GlobalData_t::DirListPkt, FM_DirListPkt_t::DirName,

FM_DirListEntry_t::EntryName, FM_DirListPkt_t::FileList, FM_DirListPkt_t::FirstFile, FM_CHILD_STAT_SLEEP_↩ FILECOUNT, FM_ChildSleepStat(), FM_DIR_LIST_PKT_ENTRIES, FM_DIR_LIST_TLM_MID, FM_GET_DIR_P↩ KT_CMD_EID, FM_GET_DIR_PKT_OS_ERR_EID, FM_GET_DIR_PKT_WARNING_EID, FM_GlobalData, FM_↩ PARENT_DIRECTORY, FM_THIS_DIRECTORY, FM_ChildQueueEntry_t::GetSizeTimeMode, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_MAX_PATH_LEN, OS_OBJECT_ID_UND↩ EFINED, OS_SUCCESS, FM_DirListPkt_t::PacketFiles, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::↩ Source2, FM_DirListPkt_t::TlmHeader, and FM_DirListPkt_t::TotalFiles.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.11 FM_ChildFileInfoCmd()

```
void FM_ChildFileInfoCmd (
            FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get File Info Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get file info command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetFileInfoCmd_t

Definition at line 847 of file fm_child.c.

References CFE_ES_CalculateCRC(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MISSION_ES_CRC_16, CFE_MISSION_ES_CRC_32, CFE_MISSION_ES_CRC_8, CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_GlobalData_t::ChildBuffer, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_FileInfoPkt_t::CRC, FM_FileInfoPkt_t::CRC_Computed, FM_ChildQueueEntry_t::FileInfoCRC, FM_GlobalData_t::FileInfoPkt, FM_ChildQueueEntry_t::FileInfoSize, FM_ChildQueueEntry_t::FileInfoState, FM_ChildQueueEntry_t::FileInfoTime, FM_FileInfoPkt_t::Filename, FM_FileInfoPkt_t::FileSize, FM_FileInfoPkt_t::FileStatus, FM_CHILD_FILE_BLOCK_SIZE, FM_CHILD_FILE_LOOP_COUNT, FM_CHILD_FILE_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_FILE_INFO_TLM_MID, FM_GET_FILE_INFO_CMD_EID, FM_GET_FILE_INFO_OPEN_ERR_EID, FM_GET_FILE_INFO_READ_WARNING_EID, FM_GET_FILE_INFO_STATE_WARNING_EID, FM_GET_FILE_INFO_TYPE_WARNING_EID, FM_GlobalData, FM_IGNORE_CRC, FM_NAME_IS_FILE_CLOSED, FM_FileInfoPkt_t::LastModifiedTime, FM_FileInfoPkt_t::Mode, FM_ChildQueueEntry_t::Mode, OS_close(), OS_FILE_FLAG_NONE, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_OpenCreate(), OS_read(), OS_READ_ONLY, OS_SUCCESS, OS_TaskDelay(), FM_ChildQueueEntry_t::Source1, and FM_FileInfoPkt_t::TlmHeader.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.12 FM_ChildInit()

```
int32 FM_ChildInit (
            void  )
```

Child Task Initialization Function.

Description

This function is invoked during FM application startup initialization to create and initialize the FM Child Task. The purpose for the child task is to process FM application commands that take too long to execute within the main task.

Assumptions, External Events, and Notes:

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

FM_AppInit

Definition at line 61 of file fm_child.c.

References CFE_ES_CreateChildTask(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::ChildSemaphore, FM_GlobalData_t::ChildTaskID, FM↩_CHILD_INIT_CREATE_ERR_EID, FM_CHILD_INIT_QSEM_ERR_EID, FM_CHILD_INIT_SEM_ERR_EID, FM_CH↩ILD_SEM_NAME, FM_CHILD_TASK_NAME, FM_CHILD_TASK_PRIORITY, FM_CHILD_TASK_STACK_SIZE, FM↩_ChildTask(), FM_GlobalData, FM_QUEUE_SEM_NAME, OS_CountSemCreate(), OS_MAX_PATH_LEN, and OS_↩MutSemCreate().

Referenced by FM_AppInit().

Here is the call graph for this function:

12.7.3.13 FM_ChildLoop()

```
void FM_ChildLoop (
            void  )
```

Child Task Main Loop Processor Function.

Description

This function is the main loop for the FM application child task. The function waits indefinitely for the parent task to grant the handshake semaphore, which is the signal that there are fresh command arguments in the child task handshake queue. The function will remain in this loop until the child task is terminated by the CFE, or until a fatal error occurs which causes the child task to terminate itself. Fatal errors are defined as any error returned by OS_CountSemTake or if the handshake queue is empty, or if the read index for the handshake queue is invalid.

Assumptions, External Events, and Notes:

See also

FM_ChildQueueEntry_t, FM_ChildProcess

Definition at line 147 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_↩ t::ChildReadIndex, FM_GlobalData_t::ChildSemaphore, FM_CHILD_QUEUE_DEPTH, FM_CHILD_TASK_PERF_↩ ID, FM_CHILD_TERM_EMPTYQ_ERR_EID, FM_CHILD_TERM_QIDX_ERR_EID, FM_CHILD_TERM_SEM_ERR_↩ EID, FM_ChildProcess(), FM_GlobalData, OS_CountSemTake(), and OS_ERROR.

Referenced by FM_ChildTask().

Here is the call graph for this function:

12.7.3.14 FM_ChildMoveCmd()

```
void FM_ChildMoveCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Move File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a move file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_MoveFileCmd_t

Definition at line 342 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_MOVE_CMD_EID, FM_↩MOVE_OS_ERR_EID, OS_mv(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::↩Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.15 FM_ChildProcess()

```
void FM_ChildProcess (
            void  )
```

Child Task Command Queue Processor Function.

Description

This function routes control to the appropriate child task command handler. After the command handler has finished, this function then updates the queue access variables to point to the next queue entry.

Assumptions, External Events, and Notes:

See also

FM_ChildQueueEntry_t, FM_ChildTask

Definition at line 205 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::ChildReadIndex, FM_ChildQueueEntry_t::CommandCode, FM_CHILD_EXE_ERR_EID, FM_CHILD_QUEUE_DEPTH, FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_CONCAT_CC, FM_COPY_CC, FM_CREATE_DIR_CC, FM_DECOMPRESS_CC, FM_DELETE_ALL_CC, FM_DELETE_CC, FM_DELETE_DIR_CC, FM_DELETE_INT_CC, FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC, FM_GET_FILE_INFO_CC, FM_GlobalData, FM_MOVE_CC, FM_RENAME_CC, FM_SET_FILE_PERM_CC, OS_MutSemGive(), and OS_MutSemTake().

Referenced by FM_ChildLoop().

Here is the call graph for this function:

12.7.3.16 FM_ChildRenameCmd()

```
void FM_ChildRenameCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Rename File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a rename file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_RenameFileCmd_t

Definition at line 384 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_RENAME_CMD_EID, FM_RENAME_OS_ERR_EID, OS_rename(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.17 FM_ChildSetPermissionsCmd()

```
void FM_ChildSetPermissionsCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Set Permissions Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a set permissions / mode on the source1 file or directory

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_SetPermCmd_t

Definition at line 1325 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_SET_PERM_CMD_EID, F↩M_SET_PERM_OS_ERR_EID, FM_ChildQueueEntry_t::Mode, OS_chmod(), OS_SUCCESS, and FM_ChildQueue↩Entry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.7.3.18 FM_ChildSizeTimeMode()

```
int32 FM_ChildSizeTimeMode (
            const char * Filename,
            uint32 * FileSize,
            uint32 * FileTime,
            uint32 * FileMode )
```

Child Task File Size Time and Mode Utility Function.

Description

This function is invoked to query the last modify time, current size and mode (permissions) for each directory entry when processing either the Get Directory List to File or Get Directory List to Packet commands.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to the combined directory and entry names.
out	*FileSize*	Pointer to the number containing the current entry size.
out	*FileTime*	Pointer to the number containing the last modify time.
out	*FileMode*	Pointer to the value containing the mode (permissions).

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1579 of file fm_child.c.

References CFE_PSP_MemSet(), OS_FILESTAT_MODE, OS_FILESTAT_SIZE, OS_FILESTAT_TIME, OS_stat(), and OS_SUCCESS.

Referenced by FM_ChildSleepStat().

Here is the call graph for this function:

12.7.3.19 FM_ChildSleepStat()

```
void FM_ChildSleepStat (
            const char * Filename,
            FM_DirListEntry_t * DirListData,
            int32 * FilesTillSleep,
            bool GetSizeTimeMode )
```

Child Task Sleep and Stat Utility Function.

Description

This function is invoked to query the last modify time, current size and mode (permissions) for each directory entry when processing either the Get Directory List to File or Get Directory List to Packet commands. However it only will sleep if FM_CHILD_STAT_SLEEP_FILECOUNT reaches zero and call FM_ChildSizeTimeMode if getSizeTime↩ Mode is TRUE, otherwise this function has no effect

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to the combined directory and entry names.
out	*DirListData*	Pointer to the data containing the current entry size, last modify time, and mode
out	*FilesTillSleep*	If this is zero the function will sleep for FM_CHILD_STAT_SLEEP_MS and reset it to FM_CHILD_STAT_SLEEP_FILECOUNT. Otherwise it will subtract 1
in	*GetSizeTimeMode*	Whether this function should call FM_ChildSizeTimeMode

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1610 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, FM_DirListEntry_t::EntrySize, FM_CHILD_STAT_SLEEP↩ _FILECOUNT, FM_CHILD_STAT_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_ChildSizeTimeMode(), FM_DirList↩ Entry_t::Mode, FM_DirListEntry_t::ModifyTime, and OS_TaskDelay().

Referenced by FM_ChildDirListFileLoop(), and FM_ChildDirListPktCmd().

Here is the call graph for this function:

12.7.3.20 FM_ChildTask()

```
void FM_ChildTask (
            void  )
```

Child Task Entry Point Function.

Description

This function is the entry point for the FM application child task. The function registers with CFE as a child task, creates the semaphore to interface with the parent task and calls the child task main loop function. Should the main loop function return due to a breakdown in the interface handshake with the parent task, this function will self delete as a child task with CFE. There is no return from CFE_ES_DeleteChildTask.

Assumptions, External Events, and Notes:

See also

CFE_ES_DeleteChildTask, FM_ChildLoop

Definition at line 118 of file fm_child.c.

References CFE_ES_ExitChildTask(), CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildSemaphore, FM_CHILD_INIT_EID, FM_ChildLoop(), FM_GlobalData, and OS_OBJECT_ID_UNDEFIN↩ED.

Referenced by FM_ChildInit().

Here is the call graph for this function:

12.8 apps/fm/fsw/src/fm_child.h File Reference

```
#include "cfe.h"
#include "fm_msg.h"
```

Functions

- int32 FM_ChildInit (void)

  Child Task Initialization Function.
- void FM_ChildTask (void)

  Child Task Entry Point Function.
- void FM_ChildLoop (void)

Child Task Main Loop Processor Function.

- void FM_ChildProcess (void)

  Child Task Command Queue Processor Function.
- void FM_ChildCopyCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Copy File Command Handler.
- void FM_ChildMoveCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Move File Command Handler.
- void FM_ChildRenameCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Rename File Command Handler.
- void FM_ChildDeleteCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete File Command Handler.
- void FM_ChildDeleteAllCmd (FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete All Files Command Handler.
- void FM_ChildConcatCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Concatenate Files Command Handler.
- void FM_ChildFileInfoCmd (FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get File Info Command Handler.
- void FM_ChildCreateDirCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Create Directory Command Handler.
- void FM_ChildDeleteDirCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Delete Directory Command Handler.
- void FM_ChildDirListFileCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get Dir List to File Command Handler.
- void FM_ChildDirListPktCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Get Dir List to Packet Command Handler.
- void FM_ChildSetPermissionsCmd (const FM_ChildQueueEntry_t *CmdArgs)

  Child Task Set Permissions Command Handler.
- bool FM_ChildDirListFileInit (osal_id_t *FileHandlePtr, const char *Directory, const char *Filename)

  Child Task Get Dir List to File Initialization Function.
- void FM_ChildDirListFileLoop (osal_id_t DirId, osal_id_t FileHandle, const char *Directory, const char *DirWith↩Sep, const char *Filename, uint8 GetSizeTimeMode)

  Child Task Get Dir List to File Loop Processor Function.
- int32 FM_ChildSizeTimeMode (const char *Filename, uint32 *FileSize, uint32 *FileTime, uint32 *FileMode)

  Child Task File Size Time and Mode Utility Function.
- void FM_ChildSleepStat (const char *Filename, FM_DirListEntry_t *DirListData, int32 *FilesTillSleep, bool Get↩SizeTimeMode)

  Child Task Sleep and Stat Utility Function.

12.8.1 Detailed Description

Prototypes for child task functions.

12.8.2 Function Documentation

12.8.2.1 FM_ChildConcatCmd()

```
void FM_ChildConcatCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Concatenate Files Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a concatenate files command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_ConcatCmd_t

Definition at line 670 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildBuffer, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_CHILD_FILE_BLOCK_SIZE, FM_CHILD_FILE_LOOP_COUNT, FM_CHILD_FILE_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_CONCAT_CMD_EID, FM_CONCAT_OPEN_SRC2_ERR_EID, FM_CONCAT_OPEN_TGT_ERR_EID, FM_CONCAT_OSCPY_ERR_EID, FM_CONCAT_OSRD_ERR_EID, FM_CONCAT_OSWR_ERR_EID, FM_GlobalData, OS_close(), OS_cp(), OS_FILE_FLAG_NONE, OS_lseek(), OS_OBJECT_ID_UNDEFINED, OS_OpenCreate(), OS_read(), OS_READ_ONLY, OS_READ_WRITE, OS_remove(), OS_SEEK_END, OS_SUCCESS, OS_TaskDelay(), OS_write(), FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.2 FM_ChildCopyCmd()

```
void FM_ChildCopyCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Copy File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a copy file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_CopyFileCmd_t

Definition at line 299 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_COPY_CMD_EID, FM_COPY_OS_ERR_EID, FM_GlobalData, OS_cp(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.3 FM_ChildCreateDirCmd()

```
void FM_ChildCreateDirCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Create Directory Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a create directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_CreateDirCmd_t

Definition at line 1008 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_CREATE_DIR_CMD_EID, FM_CREATE_D↩IR_OS_ERR_EID, FM_GlobalData, OS_mkdir(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.4 FM_ChildDeleteAllCmd()

```
void FM_ChildDeleteAllCmd (
            FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete All Files Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete all files from a directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteAllCmd_t

Definition at line 471 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DECOM_CFE_ERR_EID, FM_DECOM_CMD_EID, FM_DELETE_ALL_CMD_EID, FM_DELETE_ALL_FILES_ND_WARNING_EID, FM_DELETE_ALL_OS_ERR_EID, FM_DELETE_ALL_SKIP_WARNING_EID, FM_GetFilenameState(), FM_GlobalData, FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, FM_NAME_IS_NOT_IN_USE, FM_PARENT_DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DirectoryRewind(), OS_DIRENTRY_NAME, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_remove(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.5 FM_ChildDeleteCmd()

```
void FM_ChildDeleteCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteFileCmd_t

Definition at line 426 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DELETE_CMD_EID, FM_DELETE_INT_CC, FM_DELETE_OS_ERR_EID, FM_GlobalData, OS_remove(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.6 FM_ChildDeleteDirCmd()

```
void FM_ChildDeleteDirCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Delete Directory Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a delete directory command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_DeleteDirCmd_t

Definition at line 1050 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_DELETE_DIR_CMD_EID, FM_DELETE_DIR_EMPTY_ERR_EID, FM_DELETE_OPENDIR_OS_ERR_EID, FM_DELETE_RMDIR_OS_ERR_EID, FM_GlobalData, FM_PARENT_DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_OBJECT_ID_UNDEFINED, OS_rmdir(), OS_SUCCESS, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.7 FM_ChildDirListFileCmd()

```
void FM_ChildDirListFileCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get Dir List to File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get directory listing to a file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetDirFileCmd_t

Definition at line 1129 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_ChildDirListFileInit(), FM_ChildDirListFileLoop(), FM_GET_DIR_FILE_OSOPENDIR_ERR_EID, FM_GlobalData, FM_ChildQueueEntry_t::GetSizeTimeMode, OS_close(), OS_DirectoryClose(), OS_DirectoryOpen(), OS_OBJECT_ID_UNDEFINED, OS_SUCCESS, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.8 FM_ChildDirListFileInit()

```
bool FM_ChildDirListFileInit (
            osal_id_t * FileHandlePtr,
            const char * Directory,
            const char * Filename )
```

Child Task Get Dir List to File Initialization Function.

Description

This function creates the output file and then writes both the CFE file header and a blank copy of the directory list statistics structure to the output file. At the end of the command, software will re-write the statistics structure, this time with up to date values.

Assumptions, External Events, and Notes:

Parameters

out	*FileHandlePtr*	A pointer to a file handle variable which is modified to contain the newly created output file handle.
in	*Directory*	A pointer to a buffer containing the directory name.
in	*Filename*	A pointer to a buffer containing the output filename.

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1363 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_WriteHeader(), CFE_PSP_MemSet(), FM_GlobalData_t::ChildCmdErrCounter, CFE_FS_Header::Description, FM_GlobalData_t::DirListFileStats, FM_Dir↩ListFileStats_t::DirName, FM_DIR_LIST_FILE_SUBTYPE, FM_GET_DIR_FILE_OSCREAT_ERR_EID, FM_GET_D↩IR_FILE_WRBLANK_ERR_EID, FM_GET_DIR_FILE_WRHDR_ERR_EID, FM_GlobalData, OS_close(), OS_FILE_↩FLAG_CREATE, OS_FILE_FLAG_TRUNCATE, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_Open↩Create(), OS_READ_WRITE, OS_SUCCESS, OS_write(), and CFE_FS_Header::SubType.

Referenced by FM_ChildDirListFileCmd().

Here is the call graph for this function:

12.8.2.9 FM_ChildDirListFileLoop()

```
void FM_ChildDirListFileLoop (
            osal_id_t DirId,
            osal_id_t FileHandle,
            const char * Directory,
            const char * DirWithSep,
            const char * Filename,
            uint8 GetSizeTimeMode )
```

Child Task Get Dir List to File Loop Processor Function.

Description

This function reads each directory entry, determines the last modify time size and mode for each entry, and writes the entry data to the output file.

Assumptions, External Events, and Notes:

Parameters

in	*DirId*	Directory ID, a handle used to read directory entries.
in	*FileHandle*	Output file handle.
in	*Directory*	Pointer to a buffer containing the directory name.
in	*DirWithSep*	Pointer to directory name with path separator appended.
in	*Filename*	Pointer to a buffer containing the output filename.
in	*GetSizeTimeMode*	Option to call OS_stat for size, time, mode of files

Definition at line 1447 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_Global↩Data_t::ChildCmdWarnCounter, FM_DirListFileStats_t::DirEntries, FM_GlobalData_t::DirListFileStats, FM_DirList↩Entry_t::EntryName, FM_DirListFileStats_t::FileEntries, FM_CHILD_STAT_SLEEP_FILECOUNT, FM_ChildSleep↩Stat(), FM_DIR_LIST_FILE_ENTRIES, FM_GET_DIR_FILE_CMD_EID, FM_GET_DIR_FILE_UPSTATS_ERR_EID, FM_GET_DIR_FILE_WARNING_EID, FM_GET_DIR_FILE_WRENTRY_ERR_EID, FM_GlobalData, FM_PARENT_↩DIRECTORY, FM_THIS_DIRECTORY, OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_lseek(), OS_MAX_PATH↩_LEN, OS_SEEK_SET, OS_SUCCESS, and OS_write().

Referenced by FM_ChildDirListFileCmd().

Here is the call graph for this function:

12.8.2.10 FM_ChildDirListPktCmd()

```
void FM_ChildDirListPktCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get Dir List to Packet Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get directory listing to a packet command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetDirPktCmd_t

Definition at line 1192 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_Value↩ToMsgId(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::Child↩CmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry↩_t::CommandCode, FM_ChildQueueEntry_t::DirListOffset, FM_GlobalData_t::DirListPkt, FM_DirListPkt_t::DirName,

FM_DirListEntry_t::EntryName, FM_DirListPkt_t::FileList, FM_DirListPkt_t::FirstFile, FM_CHILD_STAT_SLEEP_↩ FILECOUNT, FM_ChildSleepStat(), FM_DIR_LIST_PKT_ENTRIES, FM_DIR_LIST_TLM_MID, FM_GET_DIR_P↩ KT_CMD_EID, FM_GET_DIR_PKT_OS_ERR_EID, FM_GET_DIR_PKT_WARNING_EID, FM_GlobalData, FM_↩ PARENT_DIRECTORY, FM_THIS_DIRECTORY, FM_ChildQueueEntry_t::GetSizeTimeMode, OS_DirectoryClose(), OS_DirectoryOpen(), OS_DirectoryRead(), OS_DIRENTRY_NAME, OS_MAX_PATH_LEN, OS_OBJECT_ID_UND↩ EFINED, OS_SUCCESS, FM_DirListPkt_t::PacketFiles, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::↩ Source2, FM_DirListPkt_t::TlmHeader, and FM_DirListPkt_t::TotalFiles.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.11 FM_ChildFileInfoCmd()

```
void FM_ChildFileInfoCmd (
            FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Get File Info Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a get file info command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_GetFileInfoCmd_t

Definition at line 847 of file fm_child.c.

References CFE_ES_CalculateCRC(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MISSION_ES_CRC_16, CFE_MISSION_ES_CRC_32, CFE_MISSION_ES_CRC_8, CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_GlobalData_t::ChildBuffer, FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_FileInfoPkt_t::CRC, FM_FileInfoPkt_t::CRC_Computed, FM_ChildQueueEntry_t::FileInfoCRC, FM_GlobalData_t::FileInfoPkt, FM_ChildQueueEntry_t::FileInfoSize, FM_ChildQueueEntry_t::FileInfoState, FM_ChildQueueEntry_t::FileInfoTime, FM_FileInfoPkt_t::Filename, FM_FileInfoPkt_t::FileSize, FM_FileInfoPkt_t::FileStatus, FM_CHILD_FILE_BLOCK_SIZE, FM_CHILD_FILE_LOOP_COUNT, FM_CHILD_FILE_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_FILE_INFO_TLM_MID, FM_GET_FILE_INFO_CMD_EID, FM_GET_FILE_INFO_OPEN_ERR_EID, FM_GET_FILE_INFO_READ_WARNING_EID, FM_GET_FILE_INFO_STATE_WARNING_EID, FM_GET_FILE_INFO_TYPE_WARNING_EID, FM_GlobalData, FM_IGNORE_CRC, FM_NAME_IS_FILE_CLOSED, FM_FileInfoPkt_t::LastModifiedTime, FM_FileInfoPkt_t::Mode, FM_ChildQueueEntry_t::Mode, OS_close(), OS_FILE_FLAG_NONE, OS_MAX_PATH_LEN, OS_OBJECT_ID_UNDEFINED, OS_OpenCreate(), OS_read(), OS_READ_ONLY, OS_SUCCESS, OS_TaskDelay(), FM_ChildQueueEntry_t::Source1, and FM_FileInfoPkt_t::TlmHeader.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.12 FM_ChildInit()

```
int32 FM_ChildInit (
            void  )
```

Child Task Initialization Function.

Description

This function is invoked during FM application startup initialization to create and initialize the FM Child Task. The purpose for the child task is to process FM application commands that take too long to execute within the main task.

Assumptions, External Events, and Notes:

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

FM_AppInit

Definition at line 61 of file fm_child.c.

References CFE_ES_CreateChildTask(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::ChildSemaphore, FM_GlobalData_t::ChildTaskID, FM↩_CHILD_INIT_CREATE_ERR_EID, FM_CHILD_INIT_QSEM_ERR_EID, FM_CHILD_INIT_SEM_ERR_EID, FM_CH↩ILD_SEM_NAME, FM_CHILD_TASK_NAME, FM_CHILD_TASK_PRIORITY, FM_CHILD_TASK_STACK_SIZE, FM↩_ChildTask(), FM_GlobalData, FM_QUEUE_SEM_NAME, OS_CountSemCreate(), OS_MAX_PATH_LEN, and OS_↩MutSemCreate().

Referenced by FM_AppInit().

Here is the call graph for this function:

12.8.2.13 FM_ChildLoop()

```
void FM_ChildLoop (
            void  )
```

Child Task Main Loop Processor Function.

Description

This function is the main loop for the FM application child task. The function waits indefinitely for the parent task to grant the handshake semaphore, which is the signal that there are fresh command arguments in the child task handshake queue. The function will remain in this loop until the child task is terminated by the CFE, or until a fatal error occurs which causes the child task to terminate itself. Fatal errors are defined as any error returned by OS_CountSemTake or if the handshake queue is empty, or if the read index for the handshake queue is invalid.

Assumptions, External Events, and Notes:

See also

FM_ChildQueueEntry_t, FM_ChildProcess

Definition at line 147 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildReadIndex, FM_GlobalData_t::ChildSemaphore, FM_CHILD_QUEUE_DEPTH, FM_CHILD_TASK_PERF_ID, FM_CHILD_TERM_EMPTYQ_ERR_EID, FM_CHILD_TERM_QIDX_ERR_EID, FM_CHILD_TERM_SEM_ERR_EID, FM_ChildProcess(), FM_GlobalData, OS_CountSemTake(), and OS_ERROR.

Referenced by FM_ChildTask().

Here is the call graph for this function:

12.8.2.14 FM_ChildMoveCmd()

```
void FM_ChildMoveCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Move File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a move file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_MoveFileCmd_t

Definition at line 342 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_MOVE_CMD_EID, FM_↩MOVE_OS_ERR_EID, OS_mv(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::↩Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.15 FM_ChildProcess()

```
void FM_ChildProcess (
            void  )
```

Child Task Command Queue Processor Function.

Description

This function routes control to the appropriate child task command handler. After the command handler has finished, this function then updates the queue access variables to point to the next queue entry.

Assumptions, External Events, and Notes:

See also

FM_ChildQueueEntry_t, FM_ChildTask

Definition at line 205 of file fm_child.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::ChildReadIndex, FM_ChildQueueEntry_t::CommandCode, FM_CHILD_EXE_ERR_EID, FM_CHILD_QUEUE_DEPTH, FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildMoveCmd(), FM_ChildRenameCmd(), FM_ChildSetPermissionsCmd(), FM_CONCAT_CC, FM_COPY_CC, FM_CREATE_DIR_CC, FM_DECOMPRESS_CC, FM_DELETE_ALL_CC, FM_DELETE_CC, FM_DELETE_DIR_CC, FM_DELETE_INT_CC, FM_GET_DIR_FILE_CC, FM_GET_DIR_PKT_CC, FM_GET_FILE_INFO_CC, FM_GlobalData, FM_MOVE_CC, FM_RENAME_CC, FM_SET_FILE_PERM_CC, OS_MutSemGive(), and OS_MutSemTake().

Referenced by FM_ChildLoop().

Here is the call graph for this function:

12.8.2.16 FM_ChildRenameCmd()

```
void FM_ChildRenameCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Rename File Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a rename file command.

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_RenameFileCmd_t

Definition at line 384 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_GlobalData_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_RENAME_CMD_EID, FM_RENAME_OS_ERR_EID, OS_rename(), OS_SUCCESS, FM_ChildQueueEntry_t::Source1, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.17 FM_ChildSetPermissionsCmd()

```
void FM_ChildSetPermissionsCmd (
            const FM_ChildQueueEntry_t * CmdArgs )
```

Child Task Set Permissions Command Handler.

Description

This function is invoked when the FM child task has been granted the child task handshake semaphore and the child task command queue contains arguments that signal a set permissions / mode on the source1 file or directory

Assumptions, External Events, and Notes:

Parameters

in	*CmdArgs*	A pointer to an entry in the child task handshake command queue which contains the arguments necessary to process this command.

See also

FM_ChildQueueEntry_t, FM_SetPermCmd_t

Definition at line 1325 of file fm_child.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCurrentCC, FM_Global↩Data_t::ChildPreviousCC, FM_ChildQueueEntry_t::CommandCode, FM_GlobalData, FM_SET_PERM_CMD_EID, F↩M_SET_PERM_OS_ERR_EID, FM_ChildQueueEntry_t::Mode, OS_chmod(), OS_SUCCESS, and FM_ChildQueue↩Entry_t::Source1.

Referenced by FM_ChildProcess().

Here is the call graph for this function:

12.8.2.18 FM_ChildSizeTimeMode()

```
int32 FM_ChildSizeTimeMode (
            const char * Filename,
            uint32 * FileSize,
            uint32 * FileTime,
            uint32 * FileMode )
```

Child Task File Size Time and Mode Utility Function.

Description

This function is invoked to query the last modify time, current size and mode (permissions) for each directory entry when processing either the Get Directory List to File or Get Directory List to Packet commands.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to the combined directory and entry names.
out	*FileSize*	Pointer to the number containing the current entry size.
out	*FileTime*	Pointer to the number containing the last modify time.
out	*FileMode*	Pointer to the value containing the mode (permissions).

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1579 of file fm_child.c.

References CFE_PSP_MemSet(), OS_FILESTAT_MODE, OS_FILESTAT_SIZE, OS_FILESTAT_TIME, OS_stat(), and OS_SUCCESS.

Referenced by FM_ChildSleepStat().

Here is the call graph for this function:

12.8.2.19 FM_ChildSleepStat()

```
void FM_ChildSleepStat (
            const char * Filename,
            FM_DirListEntry_t * DirListData,
            int32 * FilesTillSleep,
            bool GetSizeTimeMode )
```

Child Task Sleep and Stat Utility Function.

Description

This function is invoked to query the last modify time, current size and mode (permissions) for each directory entry when processing either the Get Directory List to File or Get Directory List to Packet commands. However it only will sleep if FM_CHILD_STAT_SLEEP_FILECOUNT reaches zero and call FM_ChildSizeTimeMode if getSizeTime↩ Mode is TRUE, otherwise this function has no effect

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to the combined directory and entry names.
out	*DirListData*	Pointer to the data containing the current entry size, last modify time, and mode
out	*FilesTillSleep*	If this is zero the function will sleep for FM_CHILD_STAT_SLEEP_MS and reset it to FM_CHILD_STAT_SLEEP_FILECOUNT. Otherwise it will subtract 1
in	*GetSizeTimeMode*	Whether this function should call FM_ChildSizeTimeMode

Returns

Execution status, see cFE Return Code Defines and OSAL Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

Definition at line 1610 of file fm_child.c.

References CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, FM_DirListEntry_t::EntrySize, FM_CHILD_STAT_SLEEP↩ _FILECOUNT, FM_CHILD_STAT_SLEEP_MS, FM_CHILD_TASK_PERF_ID, FM_ChildSizeTimeMode(), FM_DirList↩ Entry_t::Mode, FM_DirListEntry_t::ModifyTime, and OS_TaskDelay().

Referenced by FM_ChildDirListFileLoop(), and FM_ChildDirListPktCmd().

Here is the call graph for this function:

12.8.2.20 FM_ChildTask()

```
void FM_ChildTask (
            void  )
```

Child Task Entry Point Function.

Description

This function is the entry point for the FM application child task. The function registers with CFE as a child task, creates the semaphore to interface with the parent task and calls the child task main loop function. Should the main loop function return due to a breakdown in the interface handshake with the parent task, this function will self delete as a child task with CFE. There is no return from CFE_ES_DeleteChildTask.

Assumptions, External Events, and Notes:

See also

CFE_ES_DeleteChildTask, FM_ChildLoop

Definition at line 118 of file fm_child.c.

References CFE_ES_ExitChildTask(), CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_Global↩Data_t::ChildSemaphore, FM_CHILD_INIT_EID, FM_ChildLoop(), FM_GlobalData, and OS_OBJECT_ID_UNDEFIN↩ED.

Referenced by FM_ChildInit().

Here is the call graph for this function:

12.9 apps/fm/fsw/src/fm_cmd_utils.c File Reference

```
#include "cfe.h"
#include "fm_msg.h"
#include "fm_cmd_utils.h"
#include "fm_child.h"
#include "fm_perfids.h"
#include "fm_events.h"
#include <string.h>
#include <ctype.h>
```

Functions

- bool FM_IsValidCmdPktLength (const CFE_MSG_Message_t ∗MsgPtr, size_t ExpectedLength, uint32 EventID, const char ∗CmdText)

Verify Command Packet Length Function.

- bool FM_VerifyOverwrite (uint16 Overwrite, uint32 EventID, const char ∗CmdText)

  Verify Target File Overwrite Function.
- static void LoadOpenFileData (osal_id_t ObjId, void ∗CallbackArg)
- uint32 FM_GetOpenFilesData (const FM_OpenFilesEntry_t ∗OpenFilesData)

  Get Open Files Data Function.
- static void SearchOpenFileData (osal_id_t ObjId, void ∗CallbackArg)
- uint32 FM_GetFilenameState (char ∗Filename, uint32 BufferSize, bool FileInfoCmd)

  Get Filename State Function.
- uint32 FM_VerifyNameValid (char ∗Name, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Name Function.
- bool FM_VerifyFileClosed (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File is Closed Function.
- bool FM_VerifyFileExists (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Exists Function.
- bool FM_VerifyFileNoExist (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Does Not Exist Function.
- bool FM_VerifyFileNotOpen (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Is Not Open Function.
- bool FM_VerifyDirExists (char ∗Directory, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Directory Exists Function.
- bool FM_VerifyDirNoExist (char ∗Name, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Directory Does Not Exist Function.
- bool FM_VerifyChildTask (uint32 EventID, const char ∗CmdText)

  Verify Child Task Interface Function.
- void FM_InvokeChildTask (void)

  Invoke Child Task Function.
- void FM_AppendPathSep (char ∗Directory, uint32 BufferSize)

  Append Path Separator Function.

Variables

- static uint32 OpenFileCount = 0
- static bool FileIsOpen = false

12.9.1 Detailed Description

File Manager (FM) Command Utility Functions

Provides file manager utility function definitions for processing file manager commands

12.9.2 Function Documentation

12.9.2.1 FM_AppendPathSep()

```
void FM_AppendPathSep (
            char * Directory,
            uint32 BufferSize )
```

Append Path Separator Function.

Description

This function appends a path separator character (slash) to a directory name in advance of combining the directory name with directory entry names to create qualified filenames. The function will only append the separator character if there is room in the buffer for another character.

Assumptions, External Events, and Notes:

Parameters

in	*Directory*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer

See also

FM_GetFilenameState

Definition at line 643 of file fm_cmd_utils.c.

Referenced by FM_DeleteAllFilesCmd(), FM_GetDirListFileCmd(), and FM_GetDirListPktCmd().

12.9.2.2 FM_GetFilenameState()

```
uint32 FM_GetFilenameState (
            char * Filename,
            uint32 BufferSize,
            bool FileInfoCmd )
```

Get Filename State Function.

Description

This function performs a series of tests on the input filename to determine first whether the filename is currently in use by the file system as a file or directory. If the caller is the Get File Info command handler, the function also stores the file size and last modified timestamp.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*FileInfoCmd*	Is this for the Get File Info command?

Returns

File state

Return values

FM_NAME_IS_INVALID	
FM_NAME_IS_NOT_IN_USE	
FM_NAME_IS_FILE_OPEN	
FM_NAME_IS_FILE_CLOSED	
FM_NAME_IS_DIRECTORY	

See also

OS_stat, OS_FDGetInfo

Definition at line 167 of file fm_cmd_utils.c.

References FileIsOpen, FM_GlobalData_t::FileStatMode, FM_GlobalData_t::FileStatSize, FM_GlobalData_t::FileStat↩Time, FM_GlobalData, FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, F↩M_NAME_IS_INVALID, FM_NAME_IS_NOT_IN_USE, OS_FILESTAT_ISDIR, OS_FILESTAT_MODE, OS_FILES↩TAT_SIZE, OS_FILESTAT_TIME, OS_ForEachObject(), OS_OBJECT_CREATOR_ANY, OS_stat(), OS_SUCCESS, and SearchOpenFileData().

Referenced by FM_ChildDeleteAllCmd(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_↩VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), and FM_VerifyNameValid().

Here is the call graph for this function:

12.9.2.3 FM_GetOpenFilesData()

```
uint32 FM_GetOpenFilesData (
            const FM_OpenFilesEntry_t * OpenFilesData )
```

Get Open Files Data Function.

Description

This function creates a list of open files

Assumptions, External Events, and Notes:

Parameters

in	*OpenFilesData*	pointer to open files data

Returns

The number of open files

See also

OS_FDGetInfo

Definition at line 125 of file fm_cmd_utils.c.

References LoadOpenFileData(), OpenFileCount, OS_ForEachObject(), and OS_OBJECT_CREATOR_ANY.

Referenced by FM_GetOpenFilesCmd(), and FM_ReportHK().

Here is the call graph for this function:

12.9.2.4 FM_InvokeChildTask()

```
void FM_InvokeChildTask (
            void  )
```

Invoke Child Task Function.

Description

This function is called after the caller has loaded the next available entry in the child task queue with the arguments for the current command. The function updates the queue access index and then verifies that the Child Task is operational. If the Child Task is operational then it is signaled via handshake semaphore to process the next command from the queue. If instead, the Child Task is not operational, the Child Task queue processor function is called directly to process the current command. The difference between the two methods is in which execution thread is active when the command is processed.

Assumptions, External Events, and Notes:

See also

OS_CountSemGive, FM_ChildProcess

Definition at line 611 of file fm_cmd_utils.c.

References FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::↩ ChildSemaphore, FM_GlobalData_t::ChildWriteIndex, FM_CHILD_QUEUE_DEPTH, FM_GlobalData, OS_Count↩ SemGive(), OS_MutSemGive(), OS_MutSemTake(), and OS_ObjectIdDefined().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), F↩ M_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfo↩ Cmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.9.2.5 FM_IsValidCmdPktLength()

```
bool FM_IsValidCmdPktLength (
            const CFE_MSG_Message_t * MsgPtr,
            size_t ExpectedLength,
            uint32 EventID,
            const char * CmdText )
```

Verify Command Packet Length Function.

Description

This function is invoked from each of the command handlers to verify the length of the command packet.

Assumptions, External Events, and Notes:

Parameters

in	*MsgPtr*	Pointer to Message
in	*ExpectedLength*	Expected packet length (command specific)
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean valid packet length response

Return values

true	Packet length valid
false	Packet length invalid

Definition at line 47 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), and CFE_MSG_GetSize().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_MoveFileCmd(), FM_NoopCmd(), FM_RenameFileCmd(), FM_ReportHK(), FM_ResetCountersCmd(), FM_SetPermissionsCmd(), and FM_SetTableStateCmd().

Here is the call graph for this function:

12.9.2.6 FM_VerifyChildTask()

```
bool FM_VerifyChildTask (
            uint32 EventID,
            const char * CmdText )
```

Verify Child Task Interface Function.

Description

This function verifies that the child task interface queue is not full and that the queue index values are within bounds.

Assumptions, External Events, and Notes:

Parameters

in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean child task queue available response

Return values

true	Child task queue available
false	Child task queue not available

Definition at line 560 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_MemSet(), FM_GlobalData_t::↩ ChildQueue, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildSemaphore, FM_GlobalData_t::Child↩ WriteIndex, FM_CHILD_BROKEN_EID_OFFSET, FM_CHILD_DISABLED_EID_OFFSET, FM_CHILD_Q_FULL_EID↩ _OFFSET, FM_CHILD_QUEUE_DEPTH, FM_GlobalData, and OS_ObjectIdDefined().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), F↩ M_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfo↩ Cmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.9.2.7 FM_VerifyDirExists()

```
bool FM_VerifyDirExists (
            char * Directory,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Directory Exists Function.

Description

This function calls the Get Filename State function and generates an error event if the state is not an existing directory.

Assumptions, External Events, and Notes:

Parameters

in	*Directory*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean directory exists response

Return values

true	Directory exists
false	Directory does not exist

See also

FM_GetFilenameState

Definition at line 468 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISFILE_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_GetDirListFileCmd(), and FM_GetDirListPktCmd().

Here is the call graph for this function:

12.9.2.8 FM_VerifyDirNoExist()

```
bool FM_VerifyDirNoExist (
            char * Name,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Directory Does Not Exist Function.

Description

This function calls the Get Filename State function and generates an error event if the state is an existing file or directory.

Assumptions, External Events, and Notes:

Parameters

in	*Name*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean directory does not exist response

Return values

true	Directory does not exist
false	Directory exists

See also

FM_GetFilenameState

Definition at line 514 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CreateDirectoryCmd().

Here is the call graph for this function:

12.9.2.9 FM_VerifyFileClosed()

```
bool FM_VerifyFileClosed (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File is Closed Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than a closed file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file closed response

Return values

true	File is closed
false	File is not closed

See also

FM_GetFilenameState

Definition at line 275 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_ISOPEN_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_ConcatFilesCmd(), FM_DeleteAllFilesCmd(), and FM_DeleteFileCmd().

Here is the call graph for this function:

12.9.2.10 FM_VerifyFileExists()

```
bool FM_VerifyFileExists (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Exists Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than an open file or a closed file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file exists response

Return values

true	File exists
false	File does not exist

See also

FM_GetFilenameState

Definition at line 326 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CopyFileCmd(), FM_MoveFileCmd(), and FM_RenameFileCmd().

Here is the call graph for this function:

12.9.2.11 FM_VerifyFileNoExist()

```
bool FM_VerifyFileNoExist (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Does Not Exist Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than the name is unused the name is not a file and is not a directory.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file does not exist response

Return values

true	File does not exist
false	File exists

See also

FM_GetFilenameState

Definition at line 372 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_EXIST_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_DeleteAllFilesCmd(), FM_MoveFileCmd(), and FM_RenameFileCmd().

Here is the call graph for this function:

12.9.2.12 FM_VerifyFileNotOpen()

```
bool FM_VerifyFileNotOpen (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Is Not Open Function.

Description

This function calls the Get Filename State function and generates an error event if the state is a directory or an open file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file not open or not in use response

Return values

true	File is not open or not in use
false	File anything other than closed or not in use

See also

FM_GetFilenameState

Definition at line 418 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_INVALID_EID_OFFSET, FM_F↩NAME_ISDIR_EID_OFFSET, FM_FNAME_ISOPEN_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_↩GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM↩_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CopyFileCmd(), FM_GetDirListFileCmd(), and FM_MoveFileCmd().

Here is the call graph for this function:

12.9.2.13 FM_VerifyNameValid()

```
uint32 FM_VerifyNameValid (
            char * Name,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Name Function.

Description

This function calls the Get Filename State function and generates an error event if the state is invalid.

Assumptions, External Events, and Notes:

Parameters

in	*Name*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

File state

Return values

FM_NAME_IS_INVALID	
FM_NAME_IS_NOT_IN_USE	
FM_NAME_IS_FILE_OPEN	

Return values

FM_NAME_IS_FILE_CLOSED	
FM_NAME_IS_DIRECTORY	

See also

FM_GetFilenameState

Definition at line 251 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GetFilenameState(), and FM_NAME_IS_I↩NVALID.

Referenced by FM_GetFileInfoCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.9.2.14 FM_VerifyOverwrite()

```
bool FM_VerifyOverwrite (
            uint16 Overwrite,
            uint32 EventID,
            const char * CmdText )
```

Verify Target File Overwrite Function.

Description

This function is invoked from the copy and move file command handlers to verify the target file overwrite argument. Acceptable values are TRUE (one) and FALSE (zero).

Assumptions, External Events, and Notes:

Parameters

in	*Overwrite*	Value being tested
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean valid overwrite response

Return values

true	Overwrite argument valid
false	Overwrite argument invalid

See also

FM_COPY_CC, FM_MOVE_CC

Definition at line 74 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by FM_CopyFileCmd(), and FM_MoveFileCmd().

Here is the call graph for this function:

12.9.2.15 LoadOpenFileData()

```
static void LoadOpenFileData (
            osal_id_t ObjId,
            void * CallbackArg )  [static]
```

Definition at line 96 of file fm_cmd_utils.c.

References OS_task_prop_t::name, OpenFileCount, OS_FDGetInfo(), OS_IdentifyObject(), OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_OBJECT_TYPE_OS_STREAM, OS_SUCCESS, OS_TaskGetInfo(), OS_file_prop_t::Path, and OS_file_prop_t::User.

Referenced by FM_GetOpenFilesData().

Here is the call graph for this function:

12.9.2.16 SearchOpenFileData()

```
static void SearchOpenFileData (
            osal_id_t ObjId,
            void * CallbackArg )  [static]
```

Definition at line 141 of file fm_cmd_utils.c.

References FileIsOpen, OS_FDGetInfo(), OS_IdentifyObject(), OS_OBJECT_TYPE_OS_STREAM, OS_SUCCESS, and OS_file_prop_t::Path.

Referenced by FM_GetFilenameState().

Here is the call graph for this function:

12.9.3 Variable Documentation

12.9.3.1 FileIsOpen

```
bool FileIsOpen = false  [static]
```

Definition at line 39 of file fm_cmd_utils.c.

Referenced by FM_GetFilenameState(), and SearchOpenFileData().

12.9.3.2 OpenFileCount

```
uint32 OpenFileCount = 0  [static]
```

Definition at line 38 of file fm_cmd_utils.c.

Referenced by FM_GetOpenFilesData(), and LoadOpenFileData().

12.10 apps/fm/fsw/src/fm_cmd_utils.h File Reference

```
#include "cfe.h"
#include "fm_msg.h"
```

Functions

- bool FM_IsValidCmdPktLength (const CFE_MSG_Message_t ∗MsgPtr, size_t ExpectedLength, uint32 EventID, const char ∗CmdText)

  Verify Command Packet Length Function.
- bool FM_VerifyOverwrite (uint16 Overwrite, uint32 EventID, const char ∗CmdText)

  Verify Target File Overwrite Function.
- uint32 FM_GetOpenFilesData (const FM_OpenFilesEntry_t ∗OpenFilesData)

  Get Open Files Data Function.
- uint32 FM_GetFilenameState (char ∗Filename, uint32 BufferSize, bool FileInfoCmd)

  Get Filename State Function.
- uint32 FM_VerifyNameValid (char ∗Name, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Name Function.
- bool FM_VerifyFileClosed (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File is Closed Function.
- bool FM_VerifyFileExists (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Exists Function.
- bool FM_VerifyFileNoExist (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Does Not Exist Function.

- bool FM_VerifyFileNotOpen (char ∗Filename, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify File Is Not Open Function.
- bool FM_VerifyDirExists (char ∗Directory, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Directory Exists Function.
- bool FM_VerifyDirNoExist (char ∗Name, uint32 BufferSize, uint32 EventID, const char ∗CmdText)

  Verify Directory Does Not Exist Function.
- bool FM_VerifyChildTask (uint32 EventID, const char ∗CmdText)

  Verify Child Task Interface Function.
- void FM_InvokeChildTask (void)

  Invoke Child Task Function.
- void FM_AppendPathSep (char ∗Directory, uint32 BufferSize)

  Append Path Separator Function.

12.10.1 Detailed Description

Specification for the CFS File Manager command utility functions.

12.10.2 Function Documentation

12.10.2.1 FM_AppendPathSep()

```
void FM_AppendPathSep (
            char * Directory,
            uint32 BufferSize )
```

Append Path Separator Function.

Description

This function appends a path separator character (slash) to a directory name in advance of combining the directory name with directory entry names to create qualified filenames. The function will only append the separator character if there is room in the buffer for another character.

Assumptions, External Events, and Notes:

Parameters

in	*Directory*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer

See also

FM_GetFilenameState

Definition at line 643 of file fm_cmd_utils.c.

Referenced by FM_DeleteAllFilesCmd(), FM_GetDirListFileCmd(), and FM_GetDirListPktCmd().

12.10.2.2 FM_GetFilenameState()

```
uint32 FM_GetFilenameState (
            char * Filename,
            uint32 BufferSize,
            bool FileInfoCmd )
```

Get Filename State Function.

Description

This function performs a series of tests on the input filename to determine first whether the filename is currently in use by the file system as a file or directory. If the caller is the Get File Info command handler, the function also stores the file size and last modified timestamp.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*FileInfoCmd*	Is this for the Get File Info command?

Returns

File state

Return values

FM_NAME_IS_INVALID	
FM_NAME_IS_NOT_IN_USE	
FM_NAME_IS_FILE_OPEN	
FM_NAME_IS_FILE_CLOSED	
FM_NAME_IS_DIRECTORY	

See also

OS_stat, OS_FDGetInfo

Definition at line 167 of file fm_cmd_utils.c.

References FileIsOpen, FM_GlobalData_t::FileStatMode, FM_GlobalData_t::FileStatSize, FM_GlobalData_t::FileStat↩Time, FM_GlobalData, FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, F↩M_NAME_IS_INVALID, FM_NAME_IS_NOT_IN_USE, OS_FILESTAT_ISDIR, OS_FILESTAT_MODE, OS_FILES↩TAT_SIZE, OS_FILESTAT_TIME, OS_ForEachObject(), OS_OBJECT_CREATOR_ANY, OS_stat(), OS_SUCCESS, and SearchOpenFileData().

Referenced by FM_ChildDeleteAllCmd(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_↩VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), and FM_VerifyNameValid().

Here is the call graph for this function:

12.10.2.3 FM_GetOpenFilesData()

```
uint32 FM_GetOpenFilesData (
            const FM_OpenFilesEntry_t * OpenFilesData )
```

Get Open Files Data Function.

Description

This function creates a list of open files

Assumptions, External Events, and Notes:

Parameters

in	*OpenFilesData*	pointer to open files data

Returns

The number of open files

See also

OS_FDGetInfo

Definition at line 125 of file fm_cmd_utils.c.

References LoadOpenFileData(), OpenFileCount, OS_ForEachObject(), and OS_OBJECT_CREATOR_ANY.

Referenced by FM_GetOpenFilesCmd(), and FM_ReportHK().

Here is the call graph for this function:

12.10.2.4 FM_InvokeChildTask()

```
void FM_InvokeChildTask (
            void  )
```

Invoke Child Task Function.

Description

This function is called after the caller has loaded the next available entry in the child task queue with the arguments for the current command. The function updates the queue access index and then verifies that the Child Task is operational. If the Child Task is operational then it is signaled via handshake semaphore to process the next command from the queue. If instead, the Child Task is not operational, the Child Task queue processor function is called directly to process the current command. The difference between the two methods is in which execution thread is active when the command is processed.

Assumptions, External Events, and Notes:

See also

OS_CountSemGive, FM_ChildProcess

Definition at line 611 of file fm_cmd_utils.c.

References FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildQueueCountSem, FM_GlobalData_t::ChildSemaphore, FM_GlobalData_t::ChildWriteIndex, FM_CHILD_QUEUE_DEPTH, FM_GlobalData, OS_CountSemGive(), OS_MutSemGive(), OS_MutSemTake(), and OS_ObjectIdDefined().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.10.2.5 FM_IsValidCmdPktLength()

```
bool FM_IsValidCmdPktLength (
            const CFE_MSG_Message_t * MsgPtr,
            size_t ExpectedLength,
            uint32 EventID,
            const char * CmdText )
```

Verify Command Packet Length Function.

Description

This function is invoked from each of the command handlers to verify the length of the command packet.

Assumptions, External Events, and Notes:

Parameters

in	*MsgPtr*	Pointer to Message
in	*ExpectedLength*	Expected packet length (command specific)
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean valid packet length response

Return values

true	Packet length valid
false	Packet length invalid

Definition at line 47 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), and CFE_MSG_GetSize().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_MoveFileCmd(), FM_NoopCmd(), FM_RenameFileCmd(), FM_ReportHK(), FM_ResetCountersCmd(), FM_SetPermissionsCmd(), and FM_SetTableStateCmd().

Here is the call graph for this function:

12.10.2.6 FM_VerifyChildTask()

```
bool FM_VerifyChildTask (
            uint32 EventID,
            const char * CmdText )
```

Verify Child Task Interface Function.

Description

This function verifies that the child task interface queue is not full and that the queue index values are within bounds.

Assumptions, External Events, and Notes:

Parameters

in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean child task queue available response

Return values

true	Child task queue available
false	Child task queue not available

Definition at line 560 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_MemSet(), FM_GlobalData_t::↩ ChildQueue, FM_GlobalData_t::ChildQueueCount, FM_GlobalData_t::ChildSemaphore, FM_GlobalData_t::Child↩ WriteIndex, FM_CHILD_BROKEN_EID_OFFSET, FM_CHILD_DISABLED_EID_OFFSET, FM_CHILD_Q_FULL_EID↩ _OFFSET, FM_CHILD_QUEUE_DEPTH, FM_GlobalData, and OS_ObjectIdDefined().

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAllFilesCmd(), F↩ M_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfo↩ Cmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.10.2.7 FM_VerifyDirExists()

```
bool FM_VerifyDirExists (
            char * Directory,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Directory Exists Function.

Description

This function calls the Get Filename State function and generates an error event if the state is not an existing directory.

Assumptions, External Events, and Notes:

Parameters

in	*Directory*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean directory exists response

Return values

true	Directory exists
false	Directory does not exist

See also

FM_GetFilenameState

Definition at line 468 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISFILE_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_GetDirListFileCmd(), and FM_GetDirListPktCmd().

Here is the call graph for this function:

12.10.2.8 FM_VerifyDirNoExist()

```
bool FM_VerifyDirNoExist (
            char * Name,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Directory Does Not Exist Function.

Description

This function calls the Get Filename State function and generates an error event if the state is an existing file or directory.

Assumptions, External Events, and Notes:

Parameters

in	*Name*	Pointer to buffer containing directory name
in	*BufferSize*	Size of directory name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean directory does not exist response

Return values

true	Directory does not exist
false	Directory exists

See also

FM_GetFilenameState

Definition at line 514 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNA↩ME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_Get↩FilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NA↩ME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CreateDirectoryCmd().

Here is the call graph for this function:

12.10.2.9 FM_VerifyFileClosed()

```
bool FM_VerifyFileClosed (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File is Closed Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than a closed file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file closed response

Return values

true	File is closed
false	File is not closed

See also

FM_GetFilenameState

Definition at line 275 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_ISOPEN_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_ConcatFilesCmd(), FM_DeleteAllFilesCmd(), and FM_DeleteFileCmd().

Here is the call graph for this function:

12.10.2.10 FM_VerifyFileExists()

```
bool FM_VerifyFileExists (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Exists Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than an open file or a closed file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing filename
in	*BufferSize*	Size of filename character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file exists response

Return values

true	File exists
false	File does not exist

See also

FM_GetFilenameState

Definition at line 326 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_DNE_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CopyFileCmd(), FM_MoveFileCmd(), and FM_RenameFileCmd().

Here is the call graph for this function:

12.10.2.11 FM_VerifyFileNoExist()

```
bool FM_VerifyFileNoExist (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Does Not Exist Function.

Description

This function calls the Get Filename State function and generates an error event if the state is anything other than the name is unused the name is not a file and is not a directory.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file does not exist response

Return values

true	File does not exist
false	File exists

See also

FM_GetFilenameState

Definition at line 372 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_EXIST_EID_OFFSET, FM_FNAME_INVALID_EID_OFFSET, FM_FNAME_ISDIR_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_DeleteAllFilesCmd(), FM_MoveFileCmd(), and FM_RenameFileCmd().

Here is the call graph for this function:

12.10.2.12 FM_VerifyFileNotOpen()

```
bool FM_VerifyFileNotOpen (
            char * Filename,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify File Is Not Open Function.

Description

This function calls the Get Filename State function and generates an error event if the state is a directory or an open file.

Assumptions, External Events, and Notes:

Parameters

in	*Filename*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean file not open or not in use response

Return values

true	File is not open or not in use
false	File anything other than closed or not in use

See also

FM_GetFilenameState

Definition at line 418 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_FNAME_INVALID_EID_OFFSET, FM_F↩NAME_ISDIR_EID_OFFSET, FM_FNAME_ISOPEN_EID_OFFSET, FM_FNAME_UNKNOWN_EID_OFFSET, FM_↩GetFilenameState(), FM_NAME_IS_DIRECTORY, FM_NAME_IS_FILE_CLOSED, FM_NAME_IS_FILE_OPEN, FM↩_NAME_IS_INVALID, and FM_NAME_IS_NOT_IN_USE.

Referenced by FM_CopyFileCmd(), FM_GetDirListFileCmd(), and FM_MoveFileCmd().

Here is the call graph for this function:

12.10.2.13 FM_VerifyNameValid()

```
uint32 FM_VerifyNameValid (
            char * Name,
            uint32 BufferSize,
            uint32 EventID,
            const char * CmdText )
```

Verify Name Function.

Description

This function calls the Get Filename State function and generates an error event if the state is invalid.

Assumptions, External Events, and Notes:

Parameters

in	*Name*	Pointer to buffer containing name
in	*BufferSize*	Size of name character buffer
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

File state

Return values

FM_NAME_IS_INVALID	
FM_NAME_IS_NOT_IN_USE	
FM_NAME_IS_FILE_OPEN	
FM_NAME_IS_FILE_CLOSED	
FM_NAME_IS_DIRECTORY	

See also

FM_GetFilenameState

Definition at line 251 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), FM_GetFilenameState(), and FM_NAME_IS_I↩NVALID.

Referenced by FM_GetFileInfoCmd(), and FM_SetPermissionsCmd().

Here is the call graph for this function:

12.10.2.14 FM_VerifyOverwrite()

```
bool FM_VerifyOverwrite (
            uint16 Overwrite,
            uint32 EventID,
            const char * CmdText )
```

Verify Target File Overwrite Function.

Description

This function is invoked from the copy and move file command handlers to verify the target file overwrite argument. Acceptable values are TRUE (one) and FALSE (zero).

Assumptions, External Events, and Notes:

Parameters

in	*Overwrite*	Value being tested
in	*EventID*	Error event ID (command specific)
in	*CmdText*	Error event text (command specific)

Returns

Boolean valid overwrite response

Return values

true	Overwrite argument valid
false	Overwrite argument invalid

See also

FM_COPY_CC, FM_MOVE_CC

Definition at line 74 of file fm_cmd_utils.c.

References CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by FM_CopyFileCmd(), and FM_MoveFileCmd().

Here is the call graph for this function:

12.11 apps/fm/fsw/src/fm_cmds.c File Reference

```
#include "cfe.h"
#include "fm_msg.h"
#include "fm_msgdefs.h"
#include "fm_msgids.h"
#include "fm_events.h"
#include "fm_app.h"
#include "fm_cmds.h"
#include "fm_cmd_utils.h"
#include "fm_perfids.h"
#include "fm_platform_cfg.h"
#include "fm_version.h"
#include "fm_verify.h"
#include <string.h>
```

Functions

- bool FM_NoopCmd (const CFE_SB_Buffer_t *BufPtr)

  Move File Command Handler Function.
- bool FM_ResetCountersCmd (const CFE_SB_Buffer_t *BufPtr)

  Reset Counters Command Handler Function.
- bool FM_CopyFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Copy File Command Handler Function.
- bool FM_MoveFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Move File Command Handler Function.
- bool FM_RenameFileCmd (const CFE_SB_Buffer_t *BufPtr)

Rename File Command Handler Function.

- bool FM_DeleteFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete File Command Handler Function.

- bool FM_DeleteAllFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete All Files Command Handler Function.

- bool FM_ConcatFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Concatenate Files Command Handler Function.

- bool FM_GetFileInfoCmd (const CFE_SB_Buffer_t *BufPtr)

  Get File Information Command Handler Function.

- bool FM_GetOpenFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Open Files List Command Handler Function.

- bool FM_CreateDirectoryCmd (const CFE_SB_Buffer_t *BufPtr)

  Create Directory Command Handler Function.

- bool FM_DeleteDirectoryCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete Directory Command Handler Function.

- bool FM_GetDirListFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Directory List to Packet Command Handler Function.

- bool FM_GetDirListPktCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Directory List to Packet Command Handler Function.

- bool FM_GetFreeSpaceCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Free Space Command Handler Function.

- bool FM_SetTableStateCmd (const CFE_SB_Buffer_t *BufPtr)

  Set Table Entry State Command Handler Function.

- bool FM_SetPermissionsCmd (const CFE_SB_Buffer_t *BufPtr)

  Set File Permissions of a file.

12.11.1 Detailed Description

File Manager (FM) Application Ground Commands

Provides functions for the execution of the FM ground commands

12.11.2 Function Documentation

12.11.2.1 FM_ConcatFilesCmd()

```
bool FM_ConcatFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Concatenate Files Command Handler Function.

Description

This function concatenates two command specified source files into the command specified target file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the first source file to the target file and then appending the second source file to the target file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_CONCAT_CC, FM_ConcatCmd_t

Definition at line 458 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_CONCAT_CC, FM_CONCAT_CHILD_BASE_EID, FM_CONCAT_PKT_ERR_EID, FM_CONCAT_SRC1↩_BASE_EID, FM_CONCAT_SRC2_BASE_EID, FM_CONCAT_TGT_BASE_EID, FM_GlobalData, FM_InvokeChild↩Task(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_VerifyFileClosed(), FM_VerifyFileNoExist(), CFE_SB↩_Msg::Msg, OS_MAX_PATH_LEN, FM_ConcatCmd_t::Source1, FM_ChildQueueEntry_t::Source1, FM_ConcatCmd↩_t::Source2, FM_ChildQueueEntry_t::Source2, FM_ConcatCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.2 FM_CopyFileCmd()

```
bool FM_CopyFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Copy File Command Handler Function.

Description

This function copies the command specified source file to the command specified target file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_COPY_CC, FM_CopyFileCmd_t

Definition at line 105 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_COPY_CC, FM_COPY_CHILD_BASE_EID, FM_COPY_OVR_ERR_EID, FM_COPY_PKT_ERR_EID, F↩M_COPY_SRC_BASE_EID, FM_COPY_TGT_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmd↩PktLength(), FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), FM_↩VerifyOverwrite(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_CopyFileCmd_t::Overwrite, FM_CopyFileCmd_t↩::Source, FM_ChildQueueEntry_t::Source1, FM_CopyFileCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.3 FM_CreateDirectoryCmd()

```
bool FM_CreateDirectoryCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Create Directory Command Handler Function.

Description

This function creates the command specified directory.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_CREATE_DIR_CC, FM_CreateDirCmd_t

Definition at line 619 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_CreateDirCmd_t::Directory, FM_CREATE_DIR_CC, FM_CREATE_DIR_CHILD_BASE_EID, FM_CREAT↩E_DIR_PKT_ERR_EID, FM_CREATE_DIR_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValid↩CmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and F↩M_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.4 FM_DeleteAllFilesCmd()

```
bool FM_DeleteAllFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete All Files Command Handler Function.

Description

This function deletes all files from the command specified directory.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory and deleting each file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_ALL_CC, FM_DeleteAllCmd_t

Definition at line 345 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_DeleteAllCmd_t::Directory, FM_AppendPathSep(), FM_DECOM_CHILD_BASE_EID, FM_DECOM_PK↩T_ERR_EID, FM_DECOM_SRC_BASE_EID, FM_DECOM_TGT_BASE_EID, FM_DECOMPRESS_CC, FM_DEL↩ETE_ALL_CC, FM_DELETE_ALL_CHILD_BASE_EID, FM_DELETE_ALL_PKT_ERR_EID, FM_DELETE_ALL_S↩RC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_↩VerifyDirExists(), FM_VerifyFileClosed(), FM_VerifyFileNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_↩DecompressCmd_t::Source, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, FM_Decompress↩Cmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.5 FM_DeleteDirectoryCmd()

```
bool FM_DeleteDirectoryCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete Directory Command Handler Function.

Description

This function deletes the command specified directory.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_DIR_CC, FM_DeleteDirCmd_t

Definition at line 666 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_DeleteDirCmd_t::Directory, FM_DELETE_DIR_CC, FM_DELETE_DIR_CHILD_BASE_EID, FM_DELET↩E_DIR_PKT_ERR_EID, FM_DELETE_DIR_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValid↩CmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirExists(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM↩_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.6 FM_DeleteFileCmd()

```
bool FM_DeleteFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete File Command Handler Function.

Description

This function deletes the command specified file.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_CC, FM_DeleteFileCmd_t

Definition at line 298 of file fm_cmds.c.

References CFE_MSG_GetFcnCode(), FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_↩ ChildQueueEntry_t::CommandCode, FM_DeleteFileCmd_t::Filename, FM_DELETE_CHILD_BASE_EID, FM_DEL↩ ETE_PKT_ERR_EID, FM_DELETE_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPkt↩ Length(), FM_VerifyChildTask(), FM_VerifyFileClosed(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_Child↩ QueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.7 FM_GetDirListFileCmd()

```
bool FM_GetDirListFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Directory List to Packet Command Handler Function.

Description

This function creates an output file and writes a listing of the command specified directory to the file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_DIR_FILE_CC, FM_GetDirFileCmd_t, FM_DirListFileStats_t, FM_DirListEntry_t

Definition at line 713 of file fm_cmds.c.

References CFE_PSP_MemCpy(), FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::CommandCode, FM_GetDirFileCmd_t::Directory, FM_GetDirFileCmd_t::Filename, FM_AppendPathSep(), FM_DIR_LIST_FILE_DEFNAME, FM_GET_DIR_FILE_CC, FM_GET_DIR_FILE_CHILD_BASE_EID, FM_GET_DIR_FILE_PKT_ERR_EID, FM_GET_DIR_FILE_SRC_BASE_EID, FM_GET_DIR_FILE_TGT_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirExists(), FM_VerifyFileNotOpen(), FM_GetDirFileCmd_t::GetSizeTimeMode, FM_ChildQueueEntry_t::GetSizeTimeMode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.8 FM_GetDirListPktCmd()

```
bool FM_GetDirListPktCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Directory List to Packet Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with the directory listing data for the command specified directory, starting at the command specified directory entry.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_DIR_PKT_CC, FM_GetDirPktCmd_t, FM_DirListPkt_t

Definition at line 793 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GetDirPktCmd_t::Directory, FM_GetDirPktCmd_t::DirListOffset, FM_ChildQueueEntry_t::DirListOffset, F↩M_AppendPathSep(), FM_GET_DIR_PKT_CC, FM_GET_DIR_PKT_CHILD_BASE_EID, FM_GET_DIR_PKT_PK↩T_ERR_EID, FM_GET_DIR_PKT_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPkt↩Length(), FM_VerifyChildTask(), FM_VerifyDirExists(), FM_GetDirPktCmd_t::GetSizeTimeMode, FM_ChildQueue↩Entry_t::GetSizeTimeMode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_ChildQueueEntry_t::Source1, and FM↩_ChildQueueEntry_t::Source2.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.9 FM_GetFileInfoCmd()

```
bool FM_GetFileInfoCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get File Information Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with the current information regarding the command specified file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but collecting the status data and calculating the CRC will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_FILE_INFO_CC, FM_GetFileInfoCmd_t, FM_FileInfoPkt_t

Definition at line 520 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GetFileInfoCmd_t::FileInfoCRC, FM_ChildQueueEntry_t::FileInfoCRC, FM_ChildQueueEntry_t::FileInfo↩Size, FM_ChildQueueEntry_t::FileInfoState, FM_ChildQueueEntry_t::FileInfoTime, FM_GetFileInfoCmd_t::Filename, FM_GlobalData_t::FileStatMode, FM_GlobalData_t::FileStatSize, FM_GlobalData_t::FileStatTime, FM_FILE_INFO_↩CHILD_BASE_EID, FM_GET_FILE_INFO_CC, FM_GET_FILE_INFO_PKT_ERR_EID, FM_GET_FILE_INFO_SRC_↩ERR_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_NAME_IS_INVALID, FM_Verify↩ChildTask(), FM_VerifyNameValid(), FM_ChildQueueEntry_t::Mode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.10 FM_GetFreeSpaceCmd()

```
bool FM_GetFreeSpaceCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Free Space Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with free space data for each file system listed in the FM File System Free Space Table.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_FREE_SPACE_CC, FM_GetFreeSpaceCmd_t, FM_FreeSpacePkt_t

Definition at line 852 of file fm_cmds.c.

References OS_statvfs_t::blocks_free, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_FreeSpacePkt_t::FileSys, FM_FreeSpaceTable_t::FileSys, FM_FREE_SPACE_TLM_MID, FM_GET_FREE_SPACE_CMD_EID, FM_GET_FREE_SPACE_PKT_ERR_EID, FM_GET_FREE_SPACE_TBL_ERR_EID, FM_GlobalData, FM_IsValidCmdPktLength(), FM_OS_SYS_STAT_ERR_EID, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_ENABLED, FM_FreeSpacePktEntry_t::FreeSpace, FM_GlobalData_t::FreeSpacePkt, FM_GlobalData_t::FreeSpaceTablePtr, CFE_SB_Msg::Msg, FM_FreeSpacePktEntry_t::Name, FM_TableEntry_t::Name, OS_FileSysStatVolume(), OS_MAX_PATH_LEN, OS_SUCCESS, FM_TableEntry_t::State, and FM_FreeSpacePkt_t::TlmHeader.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.11 FM_GetOpenFilesCmd()

```
bool FM_GetOpenFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Open Files List Command Handler Function.

Description

This function creates a telemetry packet and populates it with a list of the current open files.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_OPEN_FILES_CC, FM_GetOpenFilesCmd_t, FM_OpenFilesPkt_t

Definition at line 582 of file fm_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_GET_OPEN_FILES_CMD_EID, FM_GET_OPEN_FILES_↩PKT_ERR_EID, FM_GetOpenFilesData(), FM_GlobalData, FM_IsValidCmdPktLength(), FM_OPEN_FILES_TLM_M↩ID, CFE_SB_Msg::Msg, FM_OpenFilesPkt_t::NumOpenFiles, FM_OpenFilesPkt_t::OpenFilesList, FM_GlobalData_t↩::OpenFilesPkt, and FM_OpenFilesPkt_t::TlmHeader.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.12 FM_MoveFileCmd()

```
bool FM_MoveFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Move File Command Handler Function.

Description

This function moves the command specified source file to the command specified target filename.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_MOVE_CC, FM_MoveFileCmd_t

Definition at line 173 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_MOVE_CC, FM_MOVE_CHILD_B↩ASE_EID, FM_MOVE_OVR_ERR_EID, FM_MOVE_PKT_ERR_EID, FM_MOVE_SRC_BASE_EID, FM_MOVE_TG↩T_BASE_EID, FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), FM_↩VerifyOverwrite(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_MoveFileCmd_t::Overwrite, FM_MoveFileCmd_t↩::Source, FM_ChildQueueEntry_t::Source1, FM_MoveFileCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.13 FM_NoopCmd()

```
bool FM_NoopCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Move File Command Handler Function.

Description

This function generates an event that displays the application version numbers.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_NOOP_CC, FM_NoopCmd_t

Definition at line 48 of file fm_cmds.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_IsValidCmdPktLength(), FM_MAJ↩OR_VERSION, FM_MINOR_VERSION, FM_MISSION_REV, FM_NOOP_CMD_EID, FM_NOOP_PKT_ERR_EID, F↩M_REVISION, and CFE_SB_Msg::Msg.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.14 FM_RenameFileCmd()

```
bool FM_RenameFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Rename File Command Handler Function.

Description

This function renames the command specified source file to the command specified target filename.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_RENAME_CC, FM_RenameFileCmd_t

Definition at line 242 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_RENAME_CC, FM_RENAME_C↩HILD_BASE_EID, FM_RENAME_PKT_ERR_EID, FM_RENAME_SRC_BASE_EID, FM_RENAME_TGT_BASE_E↩ID, FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_RenameFileCmd_t::Source, FM_ChildQueueEntry_t::Source1, FM_RenameFileCmd_t::Target, and FM_Child↩QueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.15 FM_ResetCountersCmd()

```
bool FM_ResetCountersCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Reset Counters Command Handler Function.

Description

This function resets the FM housekeeping telemetry counters.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_RESET_CC, FM_ResetCmd_t

Definition at line 73 of file fm_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::CommandCounter, FM_GlobalData_t::CommandErrCounter, FM_GlobalData, FM_IsValidCmdPktLength(), FM_RESET_CMD_EID, FM_RESET_PKT_ERR_EID, and CFE_SB_Msg::Msg.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.16 FM_SetPermissionsCmd()

```
bool FM_SetPermissionsCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Set File Permissions of a file.

Description

This function is a direct call to OS_chmod to set the file access

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_SET_FILE_PERM_CC, FM_SetPermCmd_t, FM_SET_PERM_CMD_EID, FM_SET_PERM_ERR_EID

Definition at line 995 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_SetPermCmd_t::FileName, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_N↩AME_IS_INVALID, FM_SET_FILE_PERM_CC, FM_SET_PERM_ERR_EID, FM_VerifyChildTask(), FM_VerifyName↩Valid(), FM_SetPermCmd_t::Mode, FM_ChildQueueEntry_t::Mode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.11.2.17 FM_SetTableStateCmd()

```
bool FM_SetTableStateCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Set Table Entry State Command Handler Function.

Description

This function modifies the enable/disable state for a single entry in the File System Free Space Table.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_SET_TABLE_STATE_CC, FM_SetTableStateCmd_t, FM_TableEntry_t

Definition at line 925 of file fm_cmds.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE↩_TBL_Modified(), FM_FreeSpaceTable_t::FileSys, FM_GlobalData, FM_IsValidCmdPktLength(), FM_SET_TABLE_↩STATE_ARG_IDX_ERR_EID, FM_SET_TABLE_STATE_ARG_STATE_ERR_EID, FM_SET_TABLE_STATE_CMD↩_EID, FM_SET_TABLE_STATE_PKT_ERR_EID, FM_SET_TABLE_STATE_TBL_ERR_EID, FM_SET_TABLE_ST↩ATE_UNUSED_ERR_EID, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_DISABLED, FM_TABLE_ENTRY_E↩NABLED, FM_TABLE_ENTRY_UNUSED, FM_GlobalData_t::FreeSpaceTableHandle, FM_GlobalData_t::FreeSpace↩TablePtr, CFE_SB_Msg::Msg, FM_TableEntry_t::State, FM_SetTableStateCmd_t::TableEntryIndex, and FM_SetTable↩StateCmd_t::TableEntryState.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12 apps/fm/fsw/src/fm_cmds.h File Reference

```
#include "cfe.h"
```

Functions

- bool FM_NoopCmd (const CFE_SB_Buffer_t *BufPtr)

  Move File Command Handler Function.
- bool FM_ResetCountersCmd (const CFE_SB_Buffer_t *BufPtr)

  Reset Counters Command Handler Function.
- bool FM_CopyFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Copy File Command Handler Function.
- bool FM_MoveFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Move File Command Handler Function.
- bool FM_RenameFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Rename File Command Handler Function.
- bool FM_DeleteFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete File Command Handler Function.
- bool FM_DeleteAllFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete All Files Command Handler Function.
- bool FM_ConcatFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Concatenate Files Command Handler Function.
- bool FM_GetFileInfoCmd (const CFE_SB_Buffer_t *BufPtr)

  Get File Information Command Handler Function.
- bool FM_GetOpenFilesCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Open Files List Command Handler Function.
- bool FM_CreateDirectoryCmd (const CFE_SB_Buffer_t *BufPtr)

  Create Directory Command Handler Function.
- bool FM_DeleteDirectoryCmd (const CFE_SB_Buffer_t *BufPtr)

  Delete Directory Command Handler Function.
- bool FM_GetDirListFileCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Directory List to Packet Command Handler Function.
- bool FM_GetDirListPktCmd (const CFE_SB_Buffer_t *BufPtr)

Get Directory List to Packet Command Handler Function.

- bool FM_GetFreeSpaceCmd (const CFE_SB_Buffer_t *BufPtr)

  Get Free Space Command Handler Function.

- bool FM_SetTableStateCmd (const CFE_SB_Buffer_t *BufPtr)

  Set Table Entry State Command Handler Function.

- bool FM_SetPermissionsCmd (const CFE_SB_Buffer_t *BufPtr)

  Set File Permissions of a file.

12.12.1 Detailed Description

Specification for the CFS FM ground commands.

12.12.2 Function Documentation

12.12.2.1 FM_ConcatFilesCmd()

```
bool FM_ConcatFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Concatenate Files Command Handler Function.

Description

This function concatenates two command specified source files into the command specified target file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the first source file to the target file and then appending the second source file to the target file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_CONCAT_CC, FM_ConcatCmd_t

Definition at line 458 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_CONCAT_CC, FM_CONCAT_CHILD_BASE_EID, FM_CONCAT_PKT_ERR_EID, FM_CONCAT_SRC1↩_BASE_EID, FM_CONCAT_SRC2_BASE_EID, FM_CONCAT_TGT_BASE_EID, FM_GlobalData, FM_InvokeChild↩Task(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_VerifyFileClosed(), FM_VerifyFileNoExist(), CFE_SB↩_Msg::Msg, OS_MAX_PATH_LEN, FM_ConcatCmd_t::Source1, FM_ChildQueueEntry_t::Source1, FM_ConcatCmd↩_t::Source2, FM_ChildQueueEntry_t::Source2, FM_ConcatCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.2 FM_CopyFileCmd()

```
bool FM_CopyFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Copy File Command Handler Function.

Description

This function copies the command specified source file to the command specified target file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_COPY_CC, FM_CopyFileCmd_t

Definition at line 105 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_COPY_CC, FM_COPY_CHILD_BASE_EID, FM_COPY_OVR_ERR_EID, FM_COPY_PKT_ERR_EID, F↩M_COPY_SRC_BASE_EID, FM_COPY_TGT_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmd↩PktLength(), FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), FM_↩VerifyOverwrite(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_CopyFileCmd_t::Overwrite, FM_CopyFileCmd_t↩::Source, FM_ChildQueueEntry_t::Source1, FM_CopyFileCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.3 FM_CreateDirectoryCmd()

```
bool FM_CreateDirectoryCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Create Directory Command Handler Function.

Description

This function creates the command specified directory.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_CREATE_DIR_CC, FM_CreateDirCmd_t

Definition at line 619 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_CreateDirCmd_t::Directory, FM_CREATE_DIR_CC, FM_CREATE_DIR_CHILD_BASE_EID, FM_CREAT↩E_DIR_PKT_ERR_EID, FM_CREATE_DIR_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValid↩CmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and F↩M_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.4 FM_DeleteAllFilesCmd()

```
bool FM_DeleteAllFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete All Files Command Handler Function.

Description

This function deletes all files from the command specified directory.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory and deleting each file will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_ALL_CC, FM_DeleteAllCmd_t

Definition at line 345 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_DeleteAllCmd_t::Directory, FM_AppendPathSep(), FM_DECOM_CHILD_BASE_EID, FM_DECOM_PK↩T_ERR_EID, FM_DECOM_SRC_BASE_EID, FM_DECOM_TGT_BASE_EID, FM_DECOMPRESS_CC, FM_DEL↩ETE_ALL_CC, FM_DELETE_ALL_CHILD_BASE_EID, FM_DELETE_ALL_PKT_ERR_EID, FM_DELETE_ALL_S↩RC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_↩VerifyDirExists(), FM_VerifyFileClosed(), FM_VerifyFileNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_↩DecompressCmd_t::Source, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, FM_Decompress↩Cmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.5 FM_DeleteDirectoryCmd()

```
bool FM_DeleteDirectoryCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete Directory Command Handler Function.

Description

This function deletes the command specified directory.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_DIR_CC, FM_DeleteDirCmd_t

Definition at line 666 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_DeleteDirCmd_t::Directory, FM_DELETE_DIR_CC, FM_DELETE_DIR_CHILD_BASE_EID, FM_DELET↩E_DIR_PKT_ERR_EID, FM_DELETE_DIR_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValid↩CmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirExists(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM↩_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.6 FM_DeleteFileCmd()

```
bool FM_DeleteFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Delete File Command Handler Function.

Description

This function deletes the command specified file.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_DELETE_CC, FM_DeleteFileCmd_t

Definition at line 298 of file fm_cmds.c.

References CFE_MSG_GetFcnCode(), FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::CommandCode, FM_DeleteFileCmd_t::Filename, FM_DELETE_CHILD_BASE_EID, FM_DELETE_PKT_ERR_EID, FM_DELETE_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_VerifyFileClosed(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.7 FM_GetDirListFileCmd()

```
bool FM_GetDirListFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Directory List to Packet Command Handler Function.

Description

This function creates an output file and writes a listing of the command specified directory to the file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_DIR_FILE_CC, FM_GetDirFileCmd_t, FM_DirListFileStats_t, FM_DirListEntry_t

Definition at line 713 of file fm_cmds.c.

References CFE_PSP_MemCpy(), FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::CommandCode, FM_GetDirFileCmd_t::Directory, FM_GetDirFileCmd_t::Filename, FM_AppendPathSep(), FM_DIR_LIST_FILE_DEFNAME, FM_GET_DIR_FILE_CC, FM_GET_DIR_FILE_CHILD_BASE_EID, FM_GET_DIR_FILE_PKT_ERR_EID, FM_GET_DIR_FILE_SRC_BASE_EID, FM_GET_DIR_FILE_TGT_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_VerifyChildTask(), FM_VerifyDirExists(), FM_VerifyFileNotOpen(), FM_GetDirFileCmd_t::GetSizeTimeMode, FM_ChildQueueEntry_t::GetSizeTimeMode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_ChildQueueEntry_t::Source1, FM_ChildQueueEntry_t::Source2, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.8 FM_GetDirListPktCmd()

```
bool FM_GetDirListPktCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Directory List to Packet Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with the directory listing data for the command specified directory, starting at the command specified directory entry.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_DIR_PKT_CC, FM_GetDirPktCmd_t, FM_DirListPkt_t

Definition at line 793 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GetDirPktCmd_t::Directory, FM_GetDirPktCmd_t::DirListOffset, FM_ChildQueueEntry_t::DirListOffset, F↩M_AppendPathSep(), FM_GET_DIR_PKT_CC, FM_GET_DIR_PKT_CHILD_BASE_EID, FM_GET_DIR_PKT_PK↩T_ERR_EID, FM_GET_DIR_PKT_SRC_BASE_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPkt↩Length(), FM_VerifyChildTask(), FM_VerifyDirExists(), FM_GetDirPktCmd_t::GetSizeTimeMode, FM_ChildQueue↩Entry_t::GetSizeTimeMode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_ChildQueueEntry_t::Source1, and FM↩_ChildQueueEntry_t::Source2.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.9 FM_GetFileInfoCmd()

```
bool FM_GetFileInfoCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get File Information Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with the current information regarding the command specified file.

Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but collecting the status data and calculating the CRC will be performed by a lower priority child task. As such, the return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_FILE_INFO_CC, FM_GetFileInfoCmd_t, FM_FileInfoPkt_t

Definition at line 520 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GetFileInfoCmd_t::FileInfoCRC, FM_ChildQueueEntry_t::FileInfoCRC, FM_ChildQueueEntry_t::FileInfo↩Size, FM_ChildQueueEntry_t::FileInfoState, FM_ChildQueueEntry_t::FileInfoTime, FM_GetFileInfoCmd_t::Filename, FM_GlobalData_t::FileStatMode, FM_GlobalData_t::FileStatSize, FM_GlobalData_t::FileStatTime, FM_FILE_INFO_↩CHILD_BASE_EID, FM_GET_FILE_INFO_CC, FM_GET_FILE_INFO_PKT_ERR_EID, FM_GET_FILE_INFO_SRC_↩ERR_EID, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_NAME_IS_INVALID, FM_Verify↩ChildTask(), FM_VerifyNameValid(), FM_ChildQueueEntry_t::Mode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.10 FM_GetFreeSpaceCmd()

```
bool FM_GetFreeSpaceCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Free Space Command Handler Function.

Description

This function creates a telemetry packet and populates the packet with free space data for each file system listed in the FM File System Free Space Table.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_FREE_SPACE_CC, FM_GetFreeSpaceCmd_t, FM_FreeSpacePkt_t

Definition at line 852 of file fm_cmds.c.

References OS_statvfs_t::blocks_free, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_FreeSpacePkt_t::FileSys, FM_FreeSpaceTable_t::FileSys, FM_FREE_SPACE_TLM_MID, FM_GET_FREE_SPACE_CMD_EID, FM_GET_FREE_SPACE_PKT_ERR_EID, FM_GET_FREE_SPACE_TBL_ERR_EID, FM_GlobalData, FM_IsValidCmdPktLength(), FM_OS_SYS_STAT_ERR_EID, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_ENABLED, FM_FreeSpacePktEntry_t::FreeSpace, FM_GlobalData_t::FreeSpacePkt, FM_GlobalData_t::FreeSpaceTablePtr, CFE_SB_Msg::Msg, FM_FreeSpacePktEntry_t::Name, FM_TableEntry_t::Name, OS_FileSysStatVolume(), OS_MAX_PATH_LEN, OS_SUCCESS, FM_TableEntry_t::State, and FM_FreeSpacePkt_t::TlmHeader.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.11 FM_GetOpenFilesCmd()

```
bool FM_GetOpenFilesCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Get Open Files List Command Handler Function.

Description

This function creates a telemetry packet and populates it with a list of the current open files.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_GET_OPEN_FILES_CC, FM_GetOpenFilesCmd_t, FM_OpenFilesPkt_t

Definition at line 582 of file fm_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_MSG_Init(), CFE_SB_TimeStampMsg(), CFE_SB_TransmitMsg(), CFE_SB_ValueToMsgId(), FM_GET_OPEN_FILES_CMD_EID, FM_GET_OPEN_FILES_↩PKT_ERR_EID, FM_GetOpenFilesData(), FM_GlobalData, FM_IsValidCmdPktLength(), FM_OPEN_FILES_TLM_M↩ID, CFE_SB_Msg::Msg, FM_OpenFilesPkt_t::NumOpenFiles, FM_OpenFilesPkt_t::OpenFilesList, FM_GlobalData_t↩::OpenFilesPkt, and FM_OpenFilesPkt_t::TlmHeader.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.12 FM_MoveFileCmd()

```
bool FM_MoveFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Move File Command Handler Function.

Description

This function moves the command specified source file to the command specified target filename.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_MOVE_CC, FM_MoveFileCmd_t

Definition at line 173 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_MOVE_CC, FM_MOVE_CHILD_B↩ASE_EID, FM_MOVE_OVR_ERR_EID, FM_MOVE_PKT_ERR_EID, FM_MOVE_SRC_BASE_EID, FM_MOVE_TG↩T_BASE_EID, FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), FM_↩VerifyOverwrite(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_MoveFileCmd_t::Overwrite, FM_MoveFileCmd_t↩::Source, FM_ChildQueueEntry_t::Source1, FM_MoveFileCmd_t::Target, and FM_ChildQueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.13 FM_NoopCmd()

```
bool FM_NoopCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Move File Command Handler Function.

Description

This function generates an event that displays the application version numbers.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_NOOP_CC, FM_NoopCmd_t

Definition at line 48 of file fm_cmds.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), FM_IsValidCmdPktLength(), FM_MAJ↩OR_VERSION, FM_MINOR_VERSION, FM_MISSION_REV, FM_NOOP_CMD_EID, FM_NOOP_PKT_ERR_EID, F↩M_REVISION, and CFE_SB_Msg::Msg.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.14 FM_RenameFileCmd()

```
bool FM_RenameFileCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Rename File Command Handler Function.

Description

This function renames the command specified source file to the command specified target filename.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_RENAME_CC, FM_RenameFileCmd_t

Definition at line 242 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_RENAME_CC, FM_RENAME_C↩HILD_BASE_EID, FM_RENAME_PKT_ERR_EID, FM_RENAME_SRC_BASE_EID, FM_RENAME_TGT_BASE_E↩ID, FM_VerifyChildTask(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, FM_RenameFileCmd_t::Source, FM_ChildQueueEntry_t::Source1, FM_RenameFileCmd_t::Target, and FM_Child↩QueueEntry_t::Target.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.15 FM_ResetCountersCmd()

```
bool FM_ResetCountersCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Reset Counters Command Handler Function.

Description

This function resets the FM housekeeping telemetry counters.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_RESET_CC, FM_ResetCmd_t

Definition at line 73 of file fm_cmds.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), FM_GlobalData_t::ChildCmdCounter, FM_GlobalData_t::ChildCmdErrCounter, FM_GlobalData_t::ChildCmdWarnCounter, FM_GlobalData_t::CommandCounter, FM_GlobalData_t::CommandErrCounter, FM_GlobalData, FM_IsValidCmdPktLength(), FM_RESET_CMD_EID, FM_RESET_PKT_ERR_EID, and CFE_SB_Msg::Msg.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.16 FM_SetPermissionsCmd()

```
bool FM_SetPermissionsCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Set File Permissions of a file.

Description

This function is a direct call to OS_chmod to set the file access

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_SET_FILE_PERM_CC, FM_SetPermCmd_t, FM_SET_PERM_CMD_EID, FM_SET_PERM_ERR_EID

Definition at line 995 of file fm_cmds.c.

References FM_GlobalData_t::ChildQueue, FM_GlobalData_t::ChildWriteIndex, FM_ChildQueueEntry_t::Command↩Code, FM_SetPermCmd_t::FileName, FM_GlobalData, FM_InvokeChildTask(), FM_IsValidCmdPktLength(), FM_N↩AME_IS_INVALID, FM_SET_FILE_PERM_CC, FM_SET_PERM_ERR_EID, FM_VerifyChildTask(), FM_VerifyName↩Valid(), FM_SetPermCmd_t::Mode, FM_ChildQueueEntry_t::Mode, CFE_SB_Msg::Msg, OS_MAX_PATH_LEN, and FM_ChildQueueEntry_t::Source1.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.12.2.17 FM_SetTableStateCmd()

```
bool FM_SetTableStateCmd (
            const CFE_SB_Buffer_t * BufPtr )
```

Set Table Entry State Command Handler Function.

Description

This function modifies the enable/disable state for a single entry in the File System Free Space Table.

Assumptions, External Events, and Notes:

Parameters

in	*BufPtr*	Pointer to Software Bus command packet.

Returns

Boolean command success response

Return values

true	Command successful
false	Command not successful

See also

FM_SET_TABLE_STATE_CC, FM_SetTableStateCmd_t, FM_TableEntry_t

Definition at line 925 of file fm_cmds.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_TBL_Modified(), FM_FreeSpaceTable_t::FileSys, FM_GlobalData, FM_IsValidCmdPktLength(), FM_SET_TABLE_STATE_ARG_IDX_ERR_EID, FM_SET_TABLE_STATE_ARG_STATE_ERR_EID, FM_SET_TABLE_STATE_CMD_EID, FM_SET_TABLE_STATE_PKT_ERR_EID, FM_SET_TABLE_STATE_TBL_ERR_EID, FM_SET_TABLE_STATE_UNUSED_ERR_EID, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_DISABLED, FM_TABLE_ENTRY_ENABLED, FM_TABLE_ENTRY_UNUSED, FM_GlobalData_t::FreeSpaceTableHandle, FM_GlobalData_t::FreeSpaceTablePtr, CFE_SB_Msg::Msg, FM_TableEntry_t::State, FM_SetTableStateCmd_t::TableEntryIndex, and FM_SetTableStateCmd_t::TableEntryState.

Referenced by FM_ProcessCmd().

Here is the call graph for this function:

12.13 apps/fm/fsw/src/fm_defs.h File Reference

Macros

- #define FM_IGNORE_CRC 0
- #define FM_THIS_DIRECTORY "."
- #define FM_PARENT_DIRECTORY ".."
- #define FM_NAME_IS_INVALID 0
- #define FM_NAME_IS_NOT_IN_USE 1
- #define FM_NAME_IS_FILE_OPEN 2
- #define FM_NAME_IS_FILE_CLOSED 3
- #define FM_NAME_IS_DIRECTORY 4
- #define FM_TABLE_ENTRY_DISABLED 0
- #define FM_TABLE_ENTRY_ENABLED 1
- #define FM_TABLE_ENTRY_UNUSED 2

12.13.1 Detailed Description

Value definitions

12.13.2 Macro Definition Documentation

12.13.2.1 FM_IGNORE_CRC

```
#define FM_IGNORE_CRC 0
```

Definition at line 33 of file fm_defs.h.

Referenced by FM_ChildFileInfoCmd().

12.13.2.2 FM_NAME_IS_DIRECTORY

```
#define FM_NAME_IS_DIRECTORY 4
```

Definition at line 54 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_GetFilenameState(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM↩_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), and FM_VerifyFileNotOpen().

12.13.2.3 FM_NAME_IS_FILE_CLOSED

```
#define FM_NAME_IS_FILE_CLOSED 3
```

Definition at line 53 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildFileInfoCmd(), FM_GetFilenameState(), FM_VerifyDirExists(), F↩M_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), and FM_VerifyFileNot↩Open().

12.13.2.4 FM_NAME_IS_FILE_OPEN

```
#define FM_NAME_IS_FILE_OPEN 2
```

Definition at line 52 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_GetFilenameState(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM↩_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), and FM_VerifyFileNotOpen().

12.13.2.5 FM_NAME_IS_INVALID

```
#define FM_NAME_IS_INVALID 0
```

Definition at line 50 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_GetFileInfoCmd(), FM_GetFilenameState(), FM_SetPermissionsCmd(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), FM_VerifyFileNotOpen(), and FM_VerifyNameValid().

12.13.2.6 FM_NAME_IS_NOT_IN_USE

```
#define FM_NAME_IS_NOT_IN_USE 1
```

Definition at line 51 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_GetFilenameState(), FM_VerifyDirExists(), FM_VerifyDirNoExist(), FM↩_VerifyFileClosed(), FM_VerifyFileExists(), FM_VerifyFileNoExist(), and FM_VerifyFileNotOpen().

12.13.2.7 FM_PARENT_DIRECTORY

```
#define FM_PARENT_DIRECTORY ".."
```

Definition at line 42 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileLoop(), and FM_ChildDirList↩PktCmd().

12.13.2.8 FM_TABLE_ENTRY_DISABLED

```
#define FM_TABLE_ENTRY_DISABLED 0
```

Definition at line 62 of file fm_defs.h.

Referenced by FM_SetTableStateCmd(), and FM_ValidateTable().

12.13.2.9 FM_TABLE_ENTRY_ENABLED

```
#define FM_TABLE_ENTRY_ENABLED 1
```

Definition at line 63 of file fm_defs.h.

Referenced by FM_GetFreeSpaceCmd(), FM_SetTableStateCmd(), and FM_ValidateTable().

12.13.2.10 FM_TABLE_ENTRY_UNUSED

```
#define FM_TABLE_ENTRY_UNUSED 2
```

Definition at line 64 of file fm_defs.h.

Referenced by FM_SetTableStateCmd(), and FM_ValidateTable().

12.13.2.11 FM_THIS_DIRECTORY

```
#define FM_THIS_DIRECTORY "."
```

Definition at line 41 of file fm_defs.h.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileLoop(), and FM_ChildDirList↩PktCmd().

12.14 apps/fm/fsw/src/fm_events.h File Reference

Macros

- #define FM_STARTUP_EID 1

  FM Initialization Event ID.
- #define FM_STARTUP_EVENTS_ERR_EID 2

  FM Initialization Register For Event Services Failed Event ID.
- #define FM_STARTUP_CREAT_PIPE_ERR_EID 3

  FM Initialization Create SB Input Pipe Failed Event ID.
- #define FM_STARTUP_SUBSCRIB_HK_ERR_EID 4

  FM Initialization Subscribe to HK Request Failed Event ID.
- #define FM_STARTUP_SUBSCRIB_GCMD_ERR_EID 5

  FM Initialization Subscribe to FM Commands Failed Event ID.
- #define FM_STARTUP_TABLE_INIT_ERR_EID 6

  FM Initialization Register Free Space Table Failed Event ID.
- #define FM_SB_RECEIVE_ERR_EID 7

  FM Main Loop Receive from Software Bus Failed Event ID.
- #define FM_EXIT_ERR_EID 8

  FM Application Termination Event ID.
- #define FM_MID_ERR_EID 9

  FM Main Loop Message ID Invalid Event ID.
- #define FM_CC_ERR_EID 10

  FM Main Loop Command Code Invalid Event ID.
- #define FM_HK_REQ_ERR_EID 11

  FM Command Packet Length Invalid Event ID.
- #define FM_NOOP_CMD_EID 12

  FM No-op Command Event ID.
- #define FM_NOOP_PKT_ERR_EID 13

FM No-op Command Length Invalid Event ID.

- #define FM_RESET_CMD_EID 14

  FM Reset Counters Command Event ID.
- #define FM_RESET_PKT_ERR_EID 15

  FM Reset Counters Command Length Invalid Event ID.
- #define FM_COPY_CMD_EID 16

  FM Copy File Command Event ID.
- #define FM_COPY_PKT_ERR_EID 17

  FM Copy File Command Length Invalid Event ID.
- #define FM_COPY_OVR_ERR_EID 18

  FM Copy File Command Overwrite Invalid Event ID.
- #define FM_COPY_OS_ERR_EID 19

  FM Copy File Command OS Error Event ID.
- #define FM_MOVE_CMD_EID 20

  FM Move File Command Event ID.
- #define FM_MOVE_PKT_ERR_EID 21

  FM Move File Command Length Invalid Event ID.
- #define FM_MOVE_OVR_ERR_EID 22

  FM Move File Command Overwrite Invalid Event ID.
- #define FM_MOVE_OS_ERR_EID 23

  FM Move File Command OS Error Event ID.
- #define FM_RENAME_CMD_EID 24

  FM Rename File Command Event ID.
- #define FM_RENAME_PKT_ERR_EID 25

  FM Rename File Command Length Invalid Event ID.
- #define FM_RENAME_OVR_ERR_EID 26

  FM Rename File Command Overwrite Invalid Event ID.
- #define FM_RENAME_OS_ERR_EID 27

  FM Rename File Command OS Error Event ID.
- #define FM_DELETE_CMD_EID 28

  FM Delete File Command Event ID.
- #define FM_DELETE_PKT_ERR_EID 29

  FM Delete File Command Length Invalid Event ID.
- #define FM_DELETE_OS_ERR_EID 30

  FM Delete File Command OS Error Event ID.
- #define FM_DELETE_ALL_CMD_EID 31

  FM Delete All Files Command Event ID.
- #define FM_DELETE_ALL_FILES_ND_WARNING_EID 32

  FM Delete All Files Unable To Delete All Event ID.
- #define FM_DELETE_ALL_SKIP_WARNING_EID 33

  FM Delete All Files Directories Skipped Event ID.
- #define FM_DELETE_ALL_PKT_ERR_EID 34

  FM Delete All Files Command Length Invalid Event ID.
- #define FM_DELETE_ALL_OS_ERR_EID 35

  FM Delete All Files Command OS Error Event ID.
- #define FM_DECOM_CMD_EID 36

  FM Decompress File Command Event ID.

- #define FM_DECOM_PKT_ERR_EID 37

  FM Decompress File Command Length Invalid Event ID.
- #define FM_DECOM_CFE_ERR_EID 38

  FM Decompress File Decompression Failed Event ID.
- #define FM_CONCAT_CMD_EID 39

  FM Concat Files Command Event ID.
- #define FM_CONCAT_PKT_ERR_EID 40

  FM Concat Files Command Length Invalid Event ID.
- #define FM_CONCAT_OSCPY_ERR_EID 41

  FM Concat Files Copy Failed Event ID.
- #define FM_CONCAT_OPEN_SRC2_ERR_EID 42

  FM Concat Files Command Open Second Source File Failed Event ID.
- #define FM_CONCAT_OPEN_TGT_ERR_EID 43

  FM Concat Files Command Open Target File Failed Event ID.
- #define FM_CONCAT_OSRD_ERR_EID 44

  FM Concat Files Command Read Second Source File Failed Event ID.
- #define FM_CONCAT_OSWR_ERR_EID 45

  FM Concat Files Command Write Target File Failed Event ID.
- #define FM_GET_FILE_INFO_CMD_EID 46

  FM Get File Info Command Event ID.
- #define FM_GET_FILE_INFO_STATE_WARNING_EID 47

  FM Get File Info Unable To Compute CRC File State Invalid Event ID.
- #define FM_GET_FILE_INFO_TYPE_WARNING_EID 48

  FM Get File Info Unable To Compute CRC, CRC Type Invalid Event ID.
- #define FM_GET_FILE_INFO_OPEN_ERR_EID 49

  FM Get File Info Unable To Compute CRC File Open Failed Event ID.
- #define FM_GET_FILE_INFO_READ_WARNING_EID 50

  FM Get File Info Unable To Compute CRC File Read Failed Event ID.
- #define FM_GET_FILE_INFO_PKT_ERR_EID 51

  FM Get File Info Command Length Invalid Event ID.
- #define FM_GET_FILE_INFO_SRC_ERR_EID 52

  FM Get File Info Command Filename Invalid Event ID.
- #define FM_GET_OPEN_FILES_CMD_EID 53

  FM Get Open Files Command Event ID.
- #define FM_GET_OPEN_FILES_PKT_ERR_EID 54

  FM Get Open Files Command Length Invalid Event ID.
- #define FM_CREATE_DIR_CMD_EID 55

  FM Create Directory Command Event ID.
- #define FM_CREATE_DIR_PKT_ERR_EID 56

  FM Create Directory Command Length Invalid Event ID.
- #define FM_CREATE_DIR_OS_ERR_EID 57

  FM Create Directory Command OS Error Event ID.
- #define FM_DELETE_DIR_CMD_EID 58

  FM Delete Directory Command Event ID.
- #define FM_DELETE_DIR_PKT_ERR_EID 59

  FM Delete Directory Command Length Invalid Event ID.
- #define FM_DELETE_DIR_EMPTY_ERR_EID 60

FM Delete Directory Command Failed Directory Not Empty Event ID.

- #define FM_DELETE_OPENDIR_OS_ERR_EID 61

  FM Delete Directory, Direcotry Open Failed Event ID.
- #define FM_DELETE_RMDIR_OS_ERR_EID 62

  FM Delete Directory Remove Directory Failed Event ID.
- #define FM_GET_DIR_FILE_CMD_EID 63

  FM Directory List To File Command Event ID.
- #define FM_GET_DIR_FILE_PKT_ERR_EID 64

  FM Directory List To File Command Length Invalid Event ID.
- #define FM_GET_DIR_FILE_WARNING_EID 65

  FM Directory List To File Command Combined Path and Name Too Long Event ID.
- #define FM_GET_DIR_FILE_OSOPENDIR_ERR_EID 66

  FM Directory List To File Directory Open Failed Event ID.
- #define FM_GET_DIR_FILE_WRBLANK_ERR_EID 67

  FM Directory List To File Write Blank Stats Failed Event ID.
- #define FM_GET_DIR_FILE_WRHDR_ERR_EID 68

  FM Directory List To File Write Header Failed Event ID.
- #define FM_GET_DIR_FILE_OSCREAT_ERR_EID 69

  FM Directory List To File Create File Failed Event ID.
- #define FM_GET_DIR_FILE_WRENTRY_ERR_EID 70

  FM Directory List To File Write Entry Failed Event ID.
- #define FM_GET_DIR_FILE_UPSTATS_ERR_EID 71

  FM Directory List To File Write Update Stats Failed Event ID.
- #define FM_GET_DIR_PKT_CMD_EID 72

  FM Directory List To Packet Command Event ID.
- #define FM_GET_DIR_PKT_WARNING_EID 73

  FM Directory List To Packet Command Directory and Entry Too Long Event ID.
- #define FM_GET_DIR_PKT_PKT_ERR_EID 74

  FM Directory List To Packet Command Length Invalid Event ID.
- #define FM_GET_DIR_PKT_OS_ERR_EID 75

  FM Directory List To Packet Directory Open Failed Event ID.
- #define FM_GET_FREE_SPACE_CMD_EID 76

  FM Get Free Space Command Event ID.
- #define FM_GET_FREE_SPACE_PKT_ERR_EID 77

  FM Get Free Space Command Length Invalid Event ID.
- #define FM_GET_FREE_SPACE_TBL_ERR_EID 78

  FM Get Free Space Table Not Loaded Event ID.
- #define FM_SET_TABLE_STATE_CMD_EID 79

  FM Set Table State Command Event ID.
- #define FM_SET_TABLE_STATE_PKT_ERR_EID 80

  FM Set Table State Command Length Invalid Event ID.
- #define FM_SET_TABLE_STATE_TBL_ERR_EID 81

  FM Set Table State Command Table Not Loaded Event ID.
- #define FM_SET_TABLE_STATE_ARG_IDX_ERR_EID 82

  FM Set Table State Command Index Invalid Event ID.
- #define FM_SET_TABLE_STATE_ARG_STATE_ERR_EID 83

  FM Set Table State Command State Invalid Event ID.

- #define FM_SET_TABLE_STATE_UNUSED_ERR_EID 84

  FM Set Table State Command Unused Entry Event ID.
- #define FM_TABLE_VERIFY_EMPTY_ERR_EID 85

  FM Free Space Table Verification Failed Empty Name Event ID.
- #define FM_TABLE_VERIFY_TOOLONG_ERR_EID 86

  FM Free Space Table Verification Failed Name Too Long Event ID.
- #define FM_TABLE_VERIFY_BAD_STATE_ERR_EID 88

  FM Free Space Table Verification Failed State Invalid Event ID.
- #define FM_CHILD_INIT_EID 89

  FM Child Task Initialization Complete Event ID.
- #define FM_CHILD_INIT_SEM_ERR_EID 90

  FM Child Task Initialization Create Semaphore Failed Event ID.
- #define FM_CHILD_INIT_QSEM_ERR_EID 91

  FM Child Task Initialization Create Queue Count Semaphore Failed Event ID.
- #define FM_CHILD_INIT_CREATE_ERR_EID 92

  FM Child Task Initialization Create Task Failed Event ID.
- #define FM_CHILD_TERM_EMPTYQ_ERR_EID 93

  FM Child Task Termination Error Empty Queue Event ID.
- #define FM_CHILD_TERM_QIDX_ERR_EID 94

  FM Child Task Termination Error Invalid Queue Index Event ID.
- #define FM_CHILD_TERM_SEM_ERR_EID 95

  FM Child Task Termination Error Semaphore Take Failed Event ID.
- #define FM_CHILD_EXE_ERR_EID 96

  FM Child Task Command Code Invalid Event ID.
- #define FM_TABLE_VERIFY_EID 97

  FM Free Space Table Validation Results Event ID.
- #define FM_SET_PERM_ERR_EID 98

  FM Set Permissions Command Length Invalid Event ID.
- #define FM_SET_PERM_CMD_EID 99

  FM Set Permissions Command Event ID.
- #define FM_SET_PERM_OS_ERR_EID 100

  FM Set Permissions Command Chmod Error Event ID.
- #define FM_TABLE_VERIFY_NULL_PTR_ERR_EID 101

  FM Free Space Table Verification Failed Null Pointer Detected.
- #define FM_SB_RECEIVE_NULL_PTR_ERR_EID 102

  FM Main Loop Software Bus Returned NULL On Success Event ID.
- #define FM_OS_SYS_STAT_ERR_EID 103

  FM Get Free Space Get File SYstem Stats Failed Event ID.
- #define FM_FNAME_INVALID_EID_OFFSET 0
- #define FM_FNAME_DNE_EID_OFFSET 1
- #define FM_FNAME_EXIST_EID_OFFSET 1 /∗ mutually exclusive with DNE ∗/
- #define FM_FNAME_ISDIR_EID_OFFSET 2
- #define FM_FNAME_ISFILE_EID_OFFSET 2 /∗ mutually exclusive with ISDIR ∗/
- #define FM_FNAME_ISOPEN_EID_OFFSET 3
- #define FM_FNAME_ISCLOSED_EID_OFFSET 4
- #define FM_FNAME_UNKNOWN_EID_OFFSET 5
- #define FM_FNAME_NUM_OFFSETS 6
- #define FM_CHILD_DISABLED_EID_OFFSET 0

- #define FM_CHILD_Q_FULL_EID_OFFSET 1
- #define FM_CHILD_BROKEN_EID_OFFSET 2
- #define FM_CHILD_NUM_OFFSETS 3
- #define FM_COPY_SRC_BASE_EID 151

  FM Child Task Copy File Source Filename Error Base ID.
- #define FM_COPY_SRC_INVALID_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Copy File Source Name Invalid Event ID.
- #define FM_COPY_SRC_DNE_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Copy File Source File Does Not Exist Event ID.
- #define FM_COPY_SRC_ISDIR_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Copy File Source File Name Is Directory Event ID.
- #define FM_COPY_SRC_UNKNOWN_ERR_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_UNKNOWN_↩EID_OFFSET)

  FM Child Task Copy File Source File Name Has Unknown State Event ID.
- #define FM_COPY_TGT_BASE_EID (FM_COPY_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Copy File Target Filename Error Base ID.
- #define FM_COPY_TGT_INVALID_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Copy File Target Filename Invalid Event ID.
- #define FM_COPY_TGT_EXIST_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)

  FM Child Task Copy File Target File Already Exists Event ID.
- #define FM_COPY_TGT_ISDIR_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Copy File Target Filename Is A Directory Event ID.
- #define FM_COPY_TGT_ISOPEN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_OF↩FSET)

  FM Child Task Copy File Target Filename Exists As Open File Event ID.
- #define FM_COPY_TGT_UNKNOWN_ERR_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_UNKNOWN_E↩ID_OFFSET)

  FM Child Task Copy File Target Filename Unknown State Event ID.
- #define FM_COPY_CHILD_BASE_EID (FM_COPY_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Copy File Child Task Error Base ID.
- #define FM_COPY_CHILD_DISABLED_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_DISABLED↩_EID_OFFSET)

  FM Child Task Copy File Child Task Disabled Event ID.
- #define FM_COPY_CHILD_FULL_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_O↩FFSET)

  FM Child Task Copy File Child Task Queue Full Event ID.
- #define FM_COPY_CHILD_BROKEN_ERR_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_BROKEN_EI↩D_OFFSET)

  FM Child Task Copy File Child Task Interface Broken Event ID.
- #define FM_MOVE_SRC_BASE_EID (FM_COPY_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Move File Source Filename Error Base ID.
- #define FM_MOVE_SRC_INVALID_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Move File Source Filename Invalid Event ID.
- #define FM_MOVE_SRC_DNE_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Move File Source File Does Not Exist Event ID.

- #define FM_MOVE_SRC_ISDIR_ERR_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Move File Source Filename Is A Directory Event ID.
- #define FM_MOVE_TGT_BASE_EID (FM_MOVE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Move File Target Filename Error Base ID.
- #define FM_MOVE_TGT_INVALID_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_INVALID_EID_O↩FFSET)

  FM Child Task Move File Target Filename Invalid Event ID.
- #define FM_MOVE_TGT_EXIST_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFS↩ET)

  FM Child Task Move File Target File Already Exists Event ID.
- #define FM_MOVE_TGT_ISDIR_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Move File Target Filename Is A Directory Event ID.
- #define FM_MOVE_TGT_ISOPEN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_ISOPEN_EID_O↩FFSET)

  FM Child Task Move File Target File Exists As An Open File Event ID.
- #define FM_MOVE_TGT_UNKNOWN_ERR_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_UNKNOWN_↩EID_OFFSET)

  FM Child Task Move File Target File Has Unknown State Event ID.
- #define FM_MOVE_CHILD_BASE_EID (FM_MOVE_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Move File Child Task Error Base ID.
- #define FM_MOVE_CHILD_DISABLED_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_DISABLED↩_EID_OFFSET)

  FM Child Task Move File Child Task Disabled Event ID.
- #define FM_MOVE_CHILD_FULL_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_O↩FFSET)

  FM Child Task Move File Child Task Queue Full Event ID.
- #define FM_MOVE_CHILD_BROKEN_ERR_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_BROKEN_EI↩D_OFFSET)

  FM Child Task Move File Child Task Inteface Broken Event ID.
- #define FM_RENAME_SRC_BASE_EID (FM_MOVE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Rename File Source Filename Error Base ID.
- #define FM_RENAME_SRC_INVALID_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_INVALID_↩EID_OFFSET)

  FM Child Task Rename File Source Filename Invalid Event ID.
- #define FM_RENAME_SRC_DNE_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_DNE_EID_OF↩FSET)

  FM Child Task Rename File Source File Does Not Exist Event ID.
- #define FM_RENAME_SRC_ISDIR_ERR_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_ISDIR_EID_↩OFFSET)

  FM Child Task Rename File Source Filename Is Directory Event ID.
- #define FM_RENAME_TGT_BASE_EID (FM_RENAME_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Rename File Target Filename Error Base ID.
- #define FM_RENAME_TGT_INVALID_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_INVALID_↩EID_OFFSET)

  FM Child Task Rename File Target Filename Invalid Event ID.
- #define FM_RENAME_TGT_EXIST_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_EXIST_EID_↩OFFSET)

  FM Child Task Rename File Target File Already Exists Event ID.

- #define FM_RENAME_TGT_ISDIR_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISDIR_EID_O↩FFSET)

  FM Child Task Rename File Target Filename Is Directory Event ID.
- #define FM_RENAME_TGT_ISOPEN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_ISOPEN_E↩ID_OFFSET)

  FM Child Task Rename File Target Filename Exists As Open File Event ID.
- #define FM_RENAME_TGT_UNKNOWN_ERR_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_UNKNO↩WN_EID_OFFSET)

  FM Child Task Rename File Target Filename Has Unknown State Event ID.
- #define FM_RENAME_CHILD_BASE_EID (FM_RENAME_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Rename File Child Task Error Base ID.
- #define FM_RENAME_CHILD_DISABLED_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_DISA↩BLED_EID_OFFSET)

  FM Child Task Rename File Child Task Disabled Event ID.
- #define FM_RENAME_CHILD_FULL_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_Q_FULL_↩EID_OFFSET)

  FM Child Task Rename File Child Task Queue Full Event ID.
- #define FM_RENAME_CHILD_BROKEN_ERR_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_BROK↩EN_EID_OFFSET)

  FM Child Task Rename File Child Task Inteface Broken Event ID.
- #define FM_DELETE_SRC_BASE_EID (FM_RENAME_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete File Source Filename Error Base ID.
- #define FM_DELETE_SRC_INVALID_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_INVALID_EI↩D_OFFSET)

  FM Child Task Delete File Source Filename Invalid Event ID.
- #define FM_DELETE_SRC_DNE_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_DNE_EID_OFF↩SET)

  FM Child Task Delete File Source File Does Not Exist Event ID.
- #define FM_DELETE_SRC_ISDIR_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISDIR_EID_O↩FFSET)

  FM Child Task Delete File Source Filename Is Directory Event ID.
- #define FM_DELETE_SRC_OPEN_ERR_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_↩OFFSET)

  FM Child Task Delete File File Is Open Event ID.
- #define FM_DELETE_CHILD_BASE_EID (FM_DELETE_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Delete File Child Task Error Base ID.
- #define FM_DELETE_CHILD_DISABLED_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_DISAB↩LED_EID_OFFSET)

  FM Child Task Delete File Child Task Disabled Event ID.
- #define FM_DELETE_CHILD_FULL_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_Q_FULL_EI↩D_OFFSET)

  FM Child Task Delete File Child Task Queue Full Event ID.
- #define FM_DELETE_CHILD_BROKEN_ERR_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_BROKE↩N_EID_OFFSET)

  FM Child Task Delete File Child Task Inteface Broken Event ID.
- #define FM_DELETE_ALL_SRC_BASE_EID (FM_DELETE_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete All Files Directory Error Base ID.
- #define FM_DELETE_ALL_SRC_INVALID_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_I↩NVALID_EID_OFFSET)

FM Child Task Delete All Files Directory Name Invalid Event ID.

- #define FM_DELETE_ALL_SRC_DNE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Delete All Files Directory Does Not Exist Event ID.

- #define FM_DELETE_ALL_SRC_FILE_ERR_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_ISFILE_EID_OFFSET)

  FM Child Task Delete All Files Directory Name Is A File Event ID.

- #define FM_DELETE_ALL_CHILD_BASE_EID (FM_DELETE_ALL_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Delete All Files Child Task Error Base ID.

- #define FM_DELETE_ALL_CHILD_DISABLED_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Delete All Files Child Task Disabled Event ID.

- #define FM_DELETE_ALL_CHILD_FULL_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Delete All Files Child Task Queue Full Event ID.

- #define FM_DELETE_ALL_CHILD_BROKEN_ERR_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Delete All Files Child Task Inteface Broken Event ID.

- #define FM_DECOM_SRC_BASE_EID (FM_DELETE_ALL_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Decompress File Source Filename Error Base ID.

- #define FM_DECOM_SRC_INVALID_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Decompress File Source Filename Invalid Event ID.

- #define FM_DECOM_SRC_DNE_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Decompress File Source Filename Does Not Exist Event ID.

- #define FM_DECOM_SRC_ISDIR_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Decompress File Source Filename Is A Directory Event ID.

- #define FM_DECOM_SRC_OPEN_ERR_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_ISOPEN_EID_OFFSET)

  FM Child Task Decompress File Source File Is Open Event ID.

- #define FM_DECOM_TGT_BASE_EID (FM_DECOM_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Decompress File Target Filename Error Base ID.

- #define FM_DECOM_TGT_INVALID_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Decompress File Target Filename Invalid Event ID.

- #define FM_DECOM_TGT_EXIST_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_EXIST_EID_OFFSET)

  FM Child Task Decompress File Target File Already Exists Event ID.

- #define FM_DECOM_TGT_ISDIR_ERR_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Decompress File Target Filename Is Directory Event ID.

- #define FM_DECOM_CHILD_BASE_EID (FM_DECOM_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Decompress File Child Task Error Base ID.

- #define FM_DECOM_CHILD_DISABLED_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Decompress File Child Task Disabled Event ID.

- #define FM_DECOM_CHILD_FULL_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID↩_OFFSET)

  FM Child Task Decompress File Child Task Queue Full Event ID.
- #define FM_DECOM_CHILD_BROKEN_ERR_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_BROKEN↩_EID_OFFSET)

  FM Child Task Decompress File Child Task Inteface Broken Event ID.
- #define FM_CONCAT_SRC1_BASE_EID (FM_DECOM_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Concat Files Source 1 Filename Error Base ID.
- #define FM_CONCAT_SRC1_INVALID_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_INVALI↩D_EID_OFFSET)

  FM Child Task Concat Files Source 1 Filename Invalid Event ID.
- #define FM_CONCAT_SRC1_DNE_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_DNE_EID_↩OFFSET)

  FM Child Task Concat Files Source 1 File Does Not Exist Event ID.
- #define FM_CONCAT_SRC1_ISDIR_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISDIR_EID↩_OFFSET)

  FM Child Task Concat Files Source 1 Filename Is Directory Event ID.
- #define FM_CONCAT_SRC1_OPEN_ERR_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_ISOPEN_↩EID_OFFSET)

  FM Child Task Concat Files Source 1 File Already Open Event ID.
- #define FM_CONCAT_SRC2_BASE_EID (FM_CONCAT_SRC1_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Source 2 Filename Error Base ID.
- #define FM_CONCAT_SRC2_INVALID_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_INVALI↩D_EID_OFFSET)

  FM Child Task Concat Files Source 2 Filename Invalid Event ID.
- #define FM_CONCAT_SRC2_DNE_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_DNE_EID_↩OFFSET)

  FM Child Task Concat Files Source 2 File Does Not Exist Event ID.
- #define FM_CONCAT_SRC2_ISDIR_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISDIR_EID↩_OFFSET)

  FM Child Task Concat Files Source 2 Filename Is Directory Event ID.
- #define FM_CONCAT_SRC2_OPEN_ERR_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_ISOPEN_↩EID_OFFSET)

  FM Child Task Concat Files Source 2 File Already Open Event ID.
- #define FM_CONCAT_TGT_BASE_EID (FM_CONCAT_SRC2_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Target Filename Error Base ID.
- #define FM_CONCAT_TGT_INVALID_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_INVALID_E↩ID_OFFSET)

  FM Child Task Concat Files Target Filename Invalid Event ID.
- #define FM_CONCAT_TGT_EXIST_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_EXIST_EID_↩OFFSET)

  FM Child Task Concat Files Target Filename Already Exists Event ID.
- #define FM_CONCAT_TGT_ISDIR_ERR_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_ISDIR_EID_O↩FFSET)

  FM Child Task Concat Files Target Filename Is Directory Event ID.
- #define FM_CONCAT_CHILD_BASE_EID (FM_CONCAT_TGT_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Concat Files Child Task Error Base ID.
- #define FM_CONCAT_CHILD_DISABLED_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_DISA↩BLED_EID_OFFSET)

FM Child Task Concat Files Child Task Disabled Event ID.

- #define FM_CONCAT_CHILD_FULL_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Concat Files Child Task Queue Full Event ID.

- #define FM_CONCAT_CHILD_BROKEN_ERR_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Concat Files Child Task Inteface Broken Event ID.

- #define FM_FILE_INFO_CHILD_BASE_EID (FM_CONCAT_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Get File Info Child Task Error Base ID.

- #define FM_FILE_INFO_CHILD_DISABLED_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Get File Info Child Task Disabled Event ID.

- #define FM_FILE_INFO_CHILD_FULL_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Get File Info Child Task Queue Full Event ID.

- #define FM_FILE_INFO_CHILD_BROKEN_ERR_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Get File Info Child Task Inteface Broken Event ID.

- #define FM_CREATE_DIR_SRC_BASE_EID (FM_FILE_INFO_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Create Directory, Directory Error Base ID.

- #define FM_CREATE_DIR_SRC_INVALID_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Create Directory Name Invalid Event ID.

- #define FM_CREATE_DIR_SRC_DNE_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_DNE_EID_OFFSET)

  FM Child Task Create Directory Name Exists As File Event ID.

- #define FM_CREATE_DIR_SRC_ISDIR_ERR_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_ISDIR_EID_OFFSET)

  FM Child Task Create Directory Already Exists Event ID.

- #define FM_CREATE_DIR_CHILD_BASE_EID (FM_CREATE_DIR_SRC_BASE_EID + FM_FNAME_NUM_OFFSETS)

  FM Child Task Create Directory Child Task Error Base ID.

- #define FM_CREATE_DIR_CHILD_DISABLED_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_DISABLED_EID_OFFSET)

  FM Child Task Create Directory Child Task Disabled Event ID.

- #define FM_CREATE_DIR_CHILD_FULL_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_Q_FULL_EID_OFFSET)

  FM Child Task Create Directory Child Task Queue Full Event ID.

- #define FM_CREATE_DIR_CHILD_BROKEN_ERR_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_BROKEN_EID_OFFSET)

  FM Child Task Create Directory Child Task Inteface Broken Event ID.

- #define FM_DELETE_DIR_SRC_BASE_EID (FM_CREATE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_OFFSETS)

  FM Child Task Delete Directory, Directory Error Base ID.

- #define FM_DELETE_DIR_SRC_INVALID_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_INVALID_EID_OFFSET)

  FM Child Task Delete Directory Name Invalid Event ID.

- #define FM_DELETE_DIR_SRC_DNE_ERR_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_DNE_↩EID_OFFSET)

  FM Child Task Delete Directory Name Exists As File Event ID.
- #define FM_DELETE_DIR_CHILD_BASE_EID (FM_DELETE_DIR_SRC_BASE_EID + FM_FNAME_NUM_O↩FFSETS)

  FM Child Task Delete Directory Child Task Error Base ID.
- #define FM_DELETE_DIR_CHILD_DISABLED_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHI↩LD_DISABLED_EID_OFFSET)

  FM Child Task Delete Directory Child Task Disabled Event ID.
- #define FM_DELETE_DIR_CHILD_FULL_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_Q↩_FULL_EID_OFFSET)

  FM Child Task Delete Directory Child Task Queue Full Event ID.
- #define FM_DELETE_DIR_CHILD_BROKEN_ERR_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD↩_BROKEN_EID_OFFSET)

  FM Child Task Delete Directory Child Task Inteface Broken Event ID.
- #define FM_GET_DIR_FILE_SRC_BASE_EID (FM_DELETE_DIR_CHILD_BASE_EID + FM_CHILD_NUM_O↩FFSETS)

  FM Child Task Directory List to File Source Filename Error Base ID.
- #define FM_GET_DIR_FILE_SRC_INVALID_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAM↩E_INVALID_EID_OFFSET)

  FM Child Task Directory List to File Directory Name Invalid Event ID.
- #define FM_GET_DIR_FILE_SRC_DNE_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_D↩NE_EID_OFFSET)

  FM Child Task Directory List to File Directory Does Not Exist Event ID.
- #define FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_I↩SFILE_EID_OFFSET)

  FM Child Task Directory List to File Directory Name Is File Event ID.
- #define FM_GET_DIR_FILE_TGT_BASE_EID (FM_GET_DIR_FILE_SRC_BASE_EID + FM_FNAME_NUM_↩OFFSETS)

  FM Child Task Directory List to File Target Error Base ID.
- #define FM_GET_DIR_FILE_TGT_INVALID_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME↩_INVALID_EID_OFFSET)

  FM Child Task Directory List to File Target Filename Invalid Event ID.
- #define FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_I↩SDIR_EID_OFFSET)

  FM Child Task Directory List to File Target Filename Is Directory Event ID.
- #define FM_GET_DIR_FILE_CHILD_BASE_EID (FM_GET_DIR_FILE_TGT_BASE_EID + FM_FNAME_NUM↩_OFFSETS)

  FM Child Task Directory List to File Child Task Error Base ID.
- #define FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_↩CHILD_DISABLED_EID_OFFSET)

  FM Child Task Directory List to File Child Task Disabled Event ID.
- #define FM_GET_DIR_FILE_CHILD_FULL_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD↩_Q_FULL_EID_OFFSET)

  FM Child Task Directory List to File Child Task Queue Full Event ID.
- #define FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_C↩HILD_BROKEN_EID_OFFSET)

  FM Child Task Directory List to File Child Task Inteface Broken Event ID.

- #define FM_GET_DIR_PKT_SRC_BASE_EID (FM_GET_DIR_FILE_CHILD_BASE_EID + FM_CHILD_NUM_↩ OFFSETS)

  FM Child Task Directory List to Packet Directory Error Base ID.
- #define FM_GET_DIR_PKT_SRC_INVALID_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME↩ _INVALID_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Name Invalid Event ID.
- #define FM_GET_DIR_PKT_SRC_DNE_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_DN↩ E_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Does Not Exist Event ID.
- #define FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_I↩ SFILE_EID_OFFSET)

  FM Child Task Directory List to Packet Directory Is File Event ID.
- #define FM_GET_DIR_PKT_CHILD_BASE_EID (FM_GET_DIR_PKT_SRC_BASE_EID + FM_FNAME_NUM↩ _OFFSETS)

  FM Child Task Directory List to Packet Child Task Error Base ID.
- #define FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_C↩ HILD_DISABLED_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Disabled Event ID.
- #define FM_GET_DIR_PKT_CHILD_FULL_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CHILD↩ _Q_FULL_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Queue Full Event ID.
- #define FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID (FM_GET_DIR_PKT_CHILD_BASE_EID + FM_CH↩ ILD_BROKEN_EID_OFFSET)

  FM Child Task Directory List to Packet Child Task Inteface Broken Event ID.

12.14.1 Detailed Description

Specification for the CFS File Manager Event Identifers.

12.15 apps/fm/fsw/src/fm_msg.h File Reference

```
#include "cfe.h"
#include "fm_platform_cfg.h"
#include "fm_defs.h"
```

Data Structures

- struct FM_HousekeepingCmd_t

  Housekeeping Request command packet structure.
- struct FM_NoopCmd_t

  No-Operation command packet structure.
- struct FM_ResetCmd_t

  Reset Counters command packet structure.
- struct FM_CopyFileCmd_t

  Copy File command packet structure.

- struct FM_MoveFileCmd_t

  Move File command packet structure.
- struct FM_RenameFileCmd_t

  Rename File command packet structure.
- struct FM_DeleteFileCmd_t

  Delete File command packet structure.
- struct FM_DeleteAllCmd_t

  Delete All command packet structure.
- struct FM_DecompressCmd_t

  Decompress File command packet structure.
- struct FM_ConcatCmd_t

  Concatenate Files command packet structure.
- struct FM_GetFileInfoCmd_t

  Get File Info command packet structure.
- struct FM_GetOpenFilesCmd_t

  Get Open Files command packet structure.
- struct FM_CreateDirCmd_t

  Create Directory command packet structure.
- struct FM_DeleteDirCmd_t

  Delete Directory command packet structure.
- struct FM_GetDirFileCmd_t

  Get DIR List to File command packet structure.
- struct FM_GetDirPktCmd_t

  Get DIR List to Packet command packet structure.
- struct FM_GetFreeSpaceCmd_t

  Get Free Space command packet structure.
- struct FM_SetTableStateCmd_t

  Set Table State command packet structure.
- struct FM_SetPermCmd_t

  Set Permissions for a file.
- struct FM_DirListEntry_t

  Get Directory Listing entry structure.
- struct FM_DirListPkt_t

  Get Directory Listing telemetry packet.
- struct FM_DirListFileStats_t

  Get Directory Listing file statistics structure.
- struct FM_FileInfoPkt_t

  Get File Info telemetry packet.
- struct FM_OpenFilesEntry_t

  Get Open Files list entry structure.
- struct FM_OpenFilesPkt_t

  Get Open Files telemetry packet.
- struct FM_FreeSpacePktEntry_t

  Get Free Space list entry structure.
- struct FM_FreeSpacePkt_t

  Get Free Space telemetry packet.
- struct FM_HousekeepingPkt_t

Housekeeping telemetry packet.

- struct FM_TableEntry_t

  Get Free Space table entry.

- struct FM_FreeSpaceTable_t

  Get Free Space table definition.

- struct FM_ChildQueueEntry_t

  Child Task Interface command queue entry structure.

- struct FM_GlobalData_t

  Application global data structure.

Variables

- FM_GlobalData_t FM_GlobalData

  File Manager global.

12.15.1 Detailed Description

Specification for the CFS FM command and telemetry messages.

12.15.2 Variable Documentation

12.15.2.1 FM_GlobalData

```
FM_GlobalData_t FM_GlobalData
```

File Manager global.

Definition at line 56 of file fm_app.c.

Referenced by FM_AcquireTablePointers(), FM_ChildConcatCmd(), FM_ChildCopyCmd(), FM_ChildCreateDirCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_Child↩DirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), FM_ChildInit(), F↩M_ChildLoop(), FM_ChildMoveCmd(), FM_ChildProcess(), FM_ChildRenameCmd(), FM_ChildSetPermissions↩Cmd(), FM_ChildTask(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectoryCmd(), FM_DeleteAll↩FilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFilenameState(), FM_GetFreeSpaceCmd(), FM_GetOpenFilesCmd(), FM_Invoke↩ChildTask(), FM_MoveFileCmd(), FM_ReleaseTablePointers(), FM_RenameFileCmd(), FM_ResetCountersCmd(), FM_SetPermissionsCmd(), FM_SetTableStateCmd(), FM_TableInit(), and FM_VerifyChildTask().

12.16 apps/fm/fsw/src/fm_msgdefs.h File Reference

Macros

- #define FM_NOOP_CC 0

  No Operation.
- #define FM_RESET_CC 1

  Reset Counters.
- #define FM_COPY_CC 2

  Copy File.
- #define FM_MOVE_CC 3

  Move File.
- #define FM_RENAME_CC 4

  Rename File.
- #define FM_DELETE_CC 5

  Delete File.
- #define FM_DELETE_ALL_CC 7

  Delete All Files.
- #define FM_DECOMPRESS_CC 8

  Decompress File.
- #define FM_CONCAT_CC 9

  Concatenate Files.
- #define FM_GET_FILE_INFO_CC 10

  Get File Information.
- #define FM_GET_OPEN_FILES_CC 11

  Get Open Files Listing.
- #define FM_CREATE_DIR_CC 12

  Create Directory.
- #define FM_DELETE_DIR_CC 13

  Remove Directory.
- #define FM_GET_DIR_FILE_CC 14

  Get Directory Listing to a File.
- #define FM_GET_DIR_PKT_CC 15

  Get Directory Listing to a Packet.
- #define FM_GET_FREE_SPACE_CC 16

  Get Free Space.
- #define FM_SET_TABLE_STATE_CC 17

  Set Free Space Table Entry State.
- #define FM_DELETE_INT_CC 18

  Delete File (internal)
- #define FM_SET_FILE_PERM_CC 19

  Set Permissions of a file.

12.16.1 Detailed Description

Specification for the CFS FM command and telemetry message macro definitions.

12.17 apps/fm/fsw/src/fm_tbl.c File Reference

```
#include "fm_platform_cfg.h"
#include "fm_msg.h"
#include "fm_tbl.h"
#include "fm_events.h"
#include <string.h>
```

Functions

- int32 FM_TableInit (void)

  Table Initialization Function.
- int32 FM_ValidateTable (FM_FreeSpaceTable_t *TablePtr)

  Table Verification Function.
- void FM_AcquireTablePointers (void)

  Acquire Table Data Pointer Function.
- void FM_ReleaseTablePointers (void)

  Release Table Data Pointer Function.

12.17.1 Detailed Description

File Manager (FM) Application Table Definitions

Provides functions for the initialization, validation, and management of the FM File System Free Space Table

12.17.2 Function Documentation

12.17.2.1 FM_AcquireTablePointers()

```
void FM_AcquireTablePointers (
            void  )
```

Acquire Table Data Pointer Function.

Description

This function is invoked to acquire a pointer to the FM file system free space table data. The pointer is maintained in the FM global data structure. Note that the table data pointer will be set to NULL if the table has not yet been successfully loaded.

Assumptions, External Events, and Notes:

See also

FM_GlobalData_t

Definition at line 189 of file fm_tbl.c.

References CFE_TBL_ERR_NEVER_LOADED, CFE_TBL_GetAddress(), CFE_TBL_Manage(), FM_GlobalData, FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::FreeSpaceTablePtr.

Referenced by FM_AppMain(), FM_ReportHK(), and FM_TableInit().

Here is the call graph for this function:

12.17.2.2 FM_ReleaseTablePointers()

```
void FM_ReleaseTablePointers (
            void  )
```

Release Table Data Pointer Function.

Description

This function is invoked to release the pointer to the FM file system free space table data. The pointer is maintained in the FM global data structure. The table data pointer must be periodically released to allow CFE Table Services an opportunity to load or dump the table without risk of interfering with users of the table data.

Assumptions, External Events, and Notes:

See also

FM_GlobalData_t

Definition at line 215 of file fm_tbl.c.

References CFE_TBL_ReleaseAddress(), FM_GlobalData, FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::FreeSpaceTablePtr.

Referenced by FM_AppMain(), and FM_ReportHK().

Here is the call graph for this function:

12.17.2.3 FM_TableInit()

```
int32 FM_TableInit (
            void  )
```

Table Initialization Function.

Description

This function is invoked during FM application startup initialization to create and initialize the FM file system free space table. The purpose for the table is to define the list of file systems for which free space must be reported.

Assumptions, External Events, and Notes:

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

/FM_AppInit

Definition at line 41 of file fm_tbl.c.

References CFE_SUCCESS, CFE_TBL_Load(), CFE_TBL_OPT_LOAD_DUMP, CFE_TBL_OPT_SNGL_BUFFER, CFE_TBL_Register(), CFE_TBL_SRC_FILE, FM_AcquireTablePointers(), FM_GlobalData, FM_TABLE_CFE_NAME, FM_TABLE_DEF_NAME, FM_ValidateTable(), FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::↩ FreeSpaceTablePtr.

Referenced by FM_AppInit().

Here is the call graph for this function:

12.17.2.4 FM_ValidateTable()

```
int32 FM_ValidateTable (
            FM_FreeSpaceTable_t * TablePtr )
```

Table Verification Function.

Description

This function is called from the CFE Table Services as part of the initial table load, and later inresponse to a Table Validate command. The function verifies that the table data is acceptable to populate the FM file system free space table.

Assumptions, External Events, and Notes:

Parameters

in	*TablePtr*	- Pointer to table data for verification.

Returns

Validation status

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
FM_TABLE_VALIDATION_ERR	Table Data Validation Error Code.

See also

/FM_AppInit

Definition at line 72 of file fm_tbl.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_↩SUCCESS, FM_FreeSpaceTable_t::FileSys, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_DISABLED, FM_T↩ABLE_ENTRY_ENABLED, FM_TABLE_ENTRY_UNUSED, FM_TABLE_VALIDATION_ERR, FM_TABLE_VERIFY_↩BAD_STATE_ERR_EID, FM_TABLE_VERIFY_EID, FM_TABLE_VERIFY_EMPTY_ERR_EID, FM_TABLE_VERIFY↩_NULL_PTR_ERR_EID, FM_TABLE_VERIFY_TOOLONG_ERR_EID, FM_TableEntry_t::Name, OS_MAX_PATH_L↩EN, and FM_TableEntry_t::State.

Referenced by FM_TableInit().

Here is the call graph for this function:

12.18 apps/fm/fsw/src/fm_tbl.h File Reference

```
#include "cfe.h"
#include "fm_msg.h"
```

Functions

- int32 FM_TableInit (void)

  Table Initialization Function.
- int32 FM_ValidateTable (FM_FreeSpaceTable_t *TablePtr)

  Table Verification Function.
- void FM_AcquireTablePointers (void)

  Acquire Table Data Pointer Function.
- void FM_ReleaseTablePointers (void)

  Release Table Data Pointer Function.

12.18.1 Detailed Description

Unit specification for the CFS File Manager table structures.

12.18.2 Function Documentation

12.18.2.1 FM_AcquireTablePointers()

```
void FM_AcquireTablePointers (
            void  )
```

Acquire Table Data Pointer Function.

Description

This function is invoked to acquire a pointer to the FM file system free space table data. The pointer is maintained in the FM global data structure. Note that the table data pointer will be set to NULL if the table has not yet been successfully loaded.

Assumptions, External Events, and Notes:

See also

FM_GlobalData_t

Definition at line 189 of file fm_tbl.c.

References CFE_TBL_ERR_NEVER_LOADED, CFE_TBL_GetAddress(), CFE_TBL_Manage(), FM_GlobalData, FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::FreeSpaceTablePtr.

Referenced by FM_AppMain(), FM_ReportHK(), and FM_TableInit().

Here is the call graph for this function:

12.18.2.2 FM_ReleaseTablePointers()

```
void FM_ReleaseTablePointers (
            void  )
```

Release Table Data Pointer Function.

Description

This function is invoked to release the pointer to the FM file system free space table data. The pointer is maintained in the FM global data structure. The table data pointer must be periodically released to allow CFE Table Services an opportunity to load or dump the table without risk of interfering with users of the table data.

Assumptions, External Events, and Notes:

See also

FM_GlobalData_t

Definition at line 215 of file fm_tbl.c.

References CFE_TBL_ReleaseAddress(), FM_GlobalData, FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::FreeSpaceTablePtr.

Referenced by FM_AppMain(), and FM_ReportHK().

Here is the call graph for this function:

12.18.2.3 FM_TableInit()

```
int32 FM_TableInit (
            void  )
```

Table Initialization Function.

Description

This function is invoked during FM application startup initialization to create and initialize the FM file system free space table. The purpose for the table is to define the list of file systems for which free space must be reported.

Assumptions, External Events, and Notes:

Returns

Execution status, see cFE Return Code Defines

Return values

CFE_SUCCESS	Successful execution.

See also

/FM_AppInit

Definition at line 41 of file fm_tbl.c.

References CFE_SUCCESS, CFE_TBL_Load(), CFE_TBL_OPT_LOAD_DUMP, CFE_TBL_OPT_SNGL_BUFFER, CFE_TBL_Register(), CFE_TBL_SRC_FILE, FM_AcquireTablePointers(), FM_GlobalData, FM_TABLE_CFE_NAME, FM_TABLE_DEF_NAME, FM_ValidateTable(), FM_GlobalData_t::FreeSpaceTableHandle, and FM_GlobalData_t::↩ FreeSpaceTablePtr.

Referenced by FM_AppInit().

Here is the call graph for this function:

12.18.2.4 FM_ValidateTable()

```
int32 FM_ValidateTable (
            FM_FreeSpaceTable_t * TablePtr )
```

Table Verification Function.

Description

This function is called from the CFE Table Services as part of the initial table load, and later inresponse to a Table Validate command. The function verifies that the table data is acceptable to populate the FM file system free space table.

Assumptions, External Events, and Notes:

Parameters

in	*TablePtr*	- Pointer to table data for verification.

Returns

Validation status

Return values

CFE_SUCCESS	Successful execution. Operation was performed successfully
FM_TABLE_VALIDATION_ERR	Table Data Validation Error Code.

See also

/FM_AppInit

Definition at line 72 of file fm_tbl.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_↩SUCCESS, FM_FreeSpaceTable_t::FileSys, FM_TABLE_ENTRY_COUNT, FM_TABLE_ENTRY_DISABLED, FM_T↩ABLE_ENTRY_ENABLED, FM_TABLE_ENTRY_UNUSED, FM_TABLE_VALIDATION_ERR, FM_TABLE_VERIFY_↩BAD_STATE_ERR_EID, FM_TABLE_VERIFY_EID, FM_TABLE_VERIFY_EMPTY_ERR_EID, FM_TABLE_VERIFY↩_NULL_PTR_ERR_EID, FM_TABLE_VERIFY_TOOLONG_ERR_EID, FM_TableEntry_t::Name, OS_MAX_PATH_L↩EN, and FM_TableEntry_t::State.

Referenced by FM_TableInit().

Here is the call graph for this function:

12.19 apps/fm/fsw/src/fm_verify.h File Reference

```
#include "cfe.h"
#include "fm_platform_cfg.h"
```

12.19.1 Detailed Description

Verification of cFS File Manager (FM) configuration parameters

12.20 apps/fm/fsw/src/fm_version.h File Reference

Macros

- #define FM_MAJOR_VERSION 2

  Major version number.
- #define FM_MINOR_VERSION 6

  Minor version number.
- #define FM_REVISION 99

  Revision number.

12.20.1 Detailed Description

Specification for the CFS FM application version label definitions

12.21 apps/fm/fsw/tables/fm_freespace.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "fm_platform_cfg.h"
#include "fm_tbl.h"
#include "fm_msg.h"
```

Functions

- static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ ((__used__))

Variables

- FM_FreeSpaceTable_t FM_FreeSpaceTable

12.21.1 Detailed Description

File Manager (FM) File System Free Space Table Data

Default table contents

12.21.2 Function Documentation

12.21.2.1 __attribute__()

```
static CFE_TBL_FileDef_t CFE_TBL_FileDef __attribute__ (
            (__used__)  )  [static]
```

12.21.3 Variable Documentation

12.21.3.1 FM_FreeSpaceTable

```
FM_FreeSpaceTable_t FM_FreeSpaceTable
```

Definition at line 58 of file fm_freespace.c.

12.22 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

  The maximum depth of OSAL queues.
- #define OS_SHELL_CMD_INPUT_FILE_NAME ""

  The name of the temporary file used to store shell commands.
- #define OS_PRINTF_CONSOLE_NAME ""

  The name of the primary console device.
- #define OS_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

12.22.1 Macro Definition Documentation

12.22.1.1 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.

12.22.1.2 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.

12.22.1.3 OS_FS_DEV_NAME_LEN

```
#define OS_FS_DEV_NAME_LEN 32
```

Device name length

Definition at line 265 of file osconfig-example.h.

12.22.1.4 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.

12.22.1.5 OS_FS_VOL_NAME_LEN

```
#define OS_FS_VOL_NAME_LEN 32
```

Volume name length

Definition at line 267 of file osconfig-example.h.

12.22.1.6 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 LoadOpenFileData().

12.22.1.7 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.

12.22.1.8 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.

12.22.1.9 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.

12.22.1.10 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.

12.22.1.11 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.

12.22.1.12 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.

12.22.1.13 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.

12.22.1.14 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.

12.22.1.15 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.

12.22.1.16 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.

12.22.1.17 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.

Referenced by FM_ChildDeleteAllCmd(), FM_ChildDirListFileInit(), FM_ChildDirListFileLoop(), FM_ChildDirListPkt↩Cmd(), FM_ChildFileInfoCmd(), FM_ChildInit(), FM_ConcatFilesCmd(), FM_CopyFileCmd(), FM_CreateDirectory↩Cmd(), FM_DeleteAllFilesCmd(), FM_DeleteDirectoryCmd(), FM_DeleteFileCmd(), FM_GetDirListFileCmd(), FM↩_GetDirListPktCmd(), FM_GetFileInfoCmd(), FM_GetFreeSpaceCmd(), FM_MoveFileCmd(), FM_RenameFileCmd(), FM_SetPermissionsCmd(), FM_ValidateTable(), and LoadOpenFileData().

12.22.1.18 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.

12.22.1.19 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.

12.22.1.20 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/FM/FM/cfe/default_config.cmake The maximum number of to support

Based on the OSAL_CONFIG_MAX_TASKS configuration option

Definition at line 44 of file osconfig-example.h.

12.22.1.21 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.

12.22.1.22 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.

12.22.1.23 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.

12.22.1.24 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.

12.22.1.25 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.

12.22.1.26 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.

12.22.1.27 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.

12.22.1.28 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.

12.22.1.29 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.

12.23 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_H↩K_MSG /∗ 0x180D ∗/

- #define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_CMD_MID_BASE + CFE_MISSION_SB_SUB_R↩ PT_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_CM↩ D_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_M↩ SG /∗ 0x0805 ∗/
- #define CFE_TIME_DIAG_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_TIME_DIAG_TL↩ M_MSG /∗ 0x0806 ∗/
- #define CFE_EVS_LONG_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_L↩ ONG_EVENT_MSG_MSG /∗ 0x0808 ∗/
- #define CFE_EVS_SHORT_EVENT_MSG_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_EVS_↩ SHORT_EVENT_MSG_MSG /∗ 0x0809 ∗/
- #define CFE_SB_STATS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_STATS_TLM↩ _MSG /∗ 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_TLM_↩ MSG /∗ 0x080C ∗/
- #define CFE_SB_ALLSUBS_TLM_MID CFE_PLATFORM_TLM_MID_BASE + CFE_MISSION_SB_ALLSUBS↩ _TLM_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_MEMS↩ TATS_TLM_MSG /∗ 0x0810 ∗/

12.23.1 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.

12.23.2 Macro Definition Documentation

12.23.2.1 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.

12.23.2.2 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.

12.23.2.3 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.

12.23.2.4 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.

12.23.2.5 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.

12.23.2.6 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.

12.23.2.7 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.

12.23.2.8 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.

12.23.2.9 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.

12.23.2.10 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.

12.23.2.11 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_MSGI↩D_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.

12.23.2.12 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_C↩MD_MID_BASE for more information

Definition at line 82 of file cpu1_msgids.h.

12.23.2.13 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.

12.23.2.14 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.

12.23.2.15 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.

12.23.2.16 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.

12.23.2.17 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.

12.23.2.18 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.

12.23.2.19 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.

12.23.2.20 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.

12.23.2.21 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.

12.23.2.22 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.

12.23.2.23 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.

12.23.2.24 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.

12.23.2.25 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.

12.23.2.26 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.

12.23.2.27 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.

12.23.2.28 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.

12.23.2.29 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.

12.23.2.30 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.

12.23.2.31 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.

12.23.2.32 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.

12.23.2.33 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.

12.23.2.34 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.

12.24 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_S↩ IZE
- #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
- #define CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE 8192
- #define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
- #define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES 512
- #define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
- #define CFE_PLATFORM_ES_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

12.24.1 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

12.24.2 Macro Definition Documentation

12.24.2.1 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.

12.24.2.2 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.

12.24.2.3 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_P↩ LATFORM_ES_APP_SCAN_RATE cycles.

Definition at line 620 of file cpu1_platform_cfg.h.

12.24.2.4 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.

12.24.2.5 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.

12.24.2.6 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.

12.24.2.7 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.

12.24.2.8 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.

12.24.2.9 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.

12.24.2.10 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.

12.24.2.11 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.

12.24.2.12 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.

12.24.2.13 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.

12.24.2.14 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.

12.24.2.15 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.

12.24.2.16 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.

12.24.2.17 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.

12.24.2.18 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.

12.24.2.19 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.

12.24.2.20 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.

12.24.2.21 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.

12.24.2.22 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.

12.24.2.23 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.

12.24.2.24 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 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.

12.24.2.25 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.

12.24.2.26 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.

12.24.2.27 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.

12.24.2.28 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.

12.24.2.29 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.

12.24.2.30 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.

12.24.2.31 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.

12.24.2.32 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.

12.24.2.33 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.

12.24.2.34 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.

12.24.2.35 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.

12.24.2.36 CFE_PLATFORM_ES_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000
```

Definition at line 1298 of file cpu1_platform_cfg.h.

12.24.2.37 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.

12.24.2.38 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.

12.24.2.39 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.

12.24.2.40 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.

12.24.2.41 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.

12.24.2.42 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.

12.24.2.43 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.

12.24.2.44 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.

12.24.2.45 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.

12.24.2.46 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.

12.24.2.47 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.

12.24.2.48 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.

12.24.2.49 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.

12.24.2.50 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.

12.24.2.51 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.

12.24.2.52 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.

12.24.2.53 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.

12.24.2.54 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.

12.24.2.55 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.

12.24.2.56 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.

12.24.2.57 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.

12.24.2.58 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.

12.24.2.59 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.

12.24.2.60 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.

12.24.2.61 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.

12.24.2.62 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.

12.24.2.63 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.

12.24.2.64 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.

12.24.2.65 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.

12.24.2.66 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.

12.24.2.67 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.

12.24.2.68 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.

12.24.2.69 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.

12.24.2.70 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.

12.24.2.71 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.

12.24.2.72 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.

12.24.2.73 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.

12.24.2.74 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.

12.24.2.75 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.

12.24.2.76 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 C↩ FE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum RAM disk sector size is system dependent and should be verified.

Definition at line 638 of file cpu1_platform_cfg.h.

12.24.2.77 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.

12.24.2.78 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.

12.24.2.79 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.

12.24.2.80 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.

12.24.2.81 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.

12.24.2.82 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.

12.24.2.83 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.

12.24.2.84 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.

12.24.2.85 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.

12.24.2.86 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.

12.24.2.87 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.

12.24.2.88 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.

12.24.2.89 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.

12.24.2.90 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.

12.24.2.91 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.

12.24.2.92 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.

12.24.2.93 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.

12.24.2.94 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.

12.24.2.95 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.

12.24.2.96 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.

12.24.2.97 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.

12.24.2.98 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.

12.24.2.99 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.

12.24.2.100 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.

12.24.2.101 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.

12.24.2.102 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.

12.24.2.103 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.

12.24.2.104 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.

12.24.2.105 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.

12.24.2.106 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.

12.24.2.107 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.

12.24.2.108 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.

12.24.2.109 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.

12.24.2.110 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.

12.24.2.111 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.

12.24.2.112 CFE_PLATFORM_SB_FILTERED_EVENT5

```
#define CFE_PLATFORM_SB_FILTERED_EVENT5 0
```

Definition at line 264 of file cpu1_platform_cfg.h.

12.24.2.113 CFE_PLATFORM_SB_FILTERED_EVENT6

```
#define CFE_PLATFORM_SB_FILTERED_EVENT6 0
```

Definition at line 267 of file cpu1_platform_cfg.h.

12.24.2.114 CFE_PLATFORM_SB_FILTERED_EVENT7

```
#define CFE_PLATFORM_SB_FILTERED_EVENT7 0
```

Definition at line 270 of file cpu1_platform_cfg.h.

12.24.2.115 CFE_PLATFORM_SB_FILTERED_EVENT8

```
#define CFE_PLATFORM_SB_FILTERED_EVENT8 0
```

Definition at line 273 of file cpu1_platform_cfg.h.

12.24.2.116 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_PLATF↩ ORM_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 0x1FFF 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.

12.24.2.117 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.

12.24.2.118 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.

12.24.2.119 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.

12.24.2.120 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.

12.24.2.121 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.

12.24.2.122 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.

12.24.2.123 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.

12.24.2.124 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.

12.24.2.125 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.

12.24.2.126 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.

12.24.2.127 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.

12.24.2.128 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.

12.24.2.129 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.

12.24.2.130 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.

12.24.2.131 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.

12.24.2.132 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.

12.24.2.133 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.

12.24.2.134 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.

12.24.2.135 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.

12.24.2.136 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.

12.24.2.137 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.

12.24.2.138 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.

12.24.2.139 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.

12.24.2.140 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.

12.24.2.141 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_T↩ ABLES.

Definition at line 1510 of file cpu1_platform_cfg.h.

12.24.2.142 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_BUF_MEMORY_BYTES.

Definition at line 1467 of file cpu1_platform_cfg.h.

12.24.2.143 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_TBL_MAX_NUM_TABLES) and should be set higher if tables are shared between applications.

Definition at line 1523 of file cpu1_platform_cfg.h.

12.24.2.144 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.

12.24.2.145 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.

12.24.2.146 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.

12.24.2.147 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_L↩OADS below when allocating memory for shared tables.

Limits

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for CFE_PLATF↩ORM_TBL_MAX_SIMULTANEOUS_LOADS number of tables to fit into CFE_PLATFORM_TBL_BUF_MEMOR↩Y_BYTES.

Definition at line 1483 of file cpu1_platform_cfg.h.

12.24.2.148 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.

12.24.2.149 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.

12.24.2.150 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.

12.24.2.151 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.

12.24.2.152 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.

12.24.2.153 CFE_PLATFORM_TBL_VALID_PRID_3

```
#define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

Definition at line 1642 of file cpu1_platform_cfg.h.

12.24.2.154 CFE_PLATFORM_TBL_VALID_PRID_4

```
#define CFE_PLATFORM_TBL_VALID_PRID_4 0
```

Definition at line 1643 of file cpu1_platform_cfg.h.

12.24.2.155 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.

12.24.2.156 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.

12.24.2.157 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.

12.24.2.158 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.

12.24.2.159 CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY 25
```

Definition at line 1157 of file cpu1_platform_cfg.h.

12.24.2.160 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.

12.24.2.161 CFE_PLATFORM_TIME_CFG_CLIENT

```
#define CFE_PLATFORM_TIME_CFG_CLIENT false
```

Definition at line 318 of file cpu1_platform_cfg.h.

12.24.2.162 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.

12.24.2.163 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_TIM↩E_CFG_CLIENT AS true. The other must be defined as false.

Definition at line 317 of file cpu1_platform_cfg.h.

12.24.2.164 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_C↩FG_SIGNAL define to true to enable tone signal commands.

Limits

Not Applicable

Definition at line 365 of file cpu1_platform_cfg.h.

12.24.2.165 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.

12.24.2.166 CFE_PLATFORM_TIME_CFG_SRC_GPS

```
#define CFE_PLATFORM_TIME_CFG_SRC_GPS false
```

Definition at line 402 of file cpu1_platform_cfg.h.

12.24.2.167 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_CF↩G_SOURCE is set to true.

Limits

1. If CFE_PLATFORM_TIME_CFG_SOURCE is set to true then one and only one of the following three external time sources can and must be set true: CFE_PLATFORM_TIME_CFG_SRC_MET, CFE_PLATFORM_TI↩ME_CFG_SRC_GPS, CFE_PLATFORM_TIME_CFG_SRC_TIME
2. Only applies if CFE_PLATFORM_TIME_CFG_SERVER is set to true.

Definition at line 401 of file cpu1_platform_cfg.h.

12.24.2.168 CFE_PLATFORM_TIME_CFG_SRC_TIME

```
#define CFE_PLATFORM_TIME_CFG_SRC_TIME false
```

Definition at line 403 of file cpu1_platform_cfg.h.

12.24.2.169 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.

12.24.2.170 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.

12.24.2.171 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.

12.24.2.172 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.

12.24.2.173 CFE_PLATFORM_TIME_MAX_DELTA_SUBS

```
#define CFE_PLATFORM_TIME_MAX_DELTA_SUBS 500000
```

Definition at line 423 of file cpu1_platform_cfg.h.

12.24.2.174 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.

12.24.2.175 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS

```
#define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
```

Definition at line 436 of file cpu1_platform_cfg.h.

12.24.2.176 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.

12.24.2.177 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.

12.24.2.178 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
```

Definition at line 1156 of file cpu1_platform_cfg.h.

12.24.2.179 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.

12.25 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_FS_FACTOR 789004800
- #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
- #define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122
- #define CFE_MISSION_ES_DEFAULT_CRC CFE_MISSION_ES_CRC_16
- #define CFE_MISSION_TBL_MAX_NAME_LENGTH 16
- #define CFE_MISSION_EVS_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_EVS_SEND_HK_MSG 9
- #define CFE_MISSION_SB_SEND_HK_MSG 11
- #define CFE_MISSION_TBL_SEND_HK_MSG 12
- #define CFE_MISSION_TIME_SEND_HK_MSG 13
- #define CFE_MISSION_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_EVS_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_EVS_LONG_EVENT_MSG_MSG 8
- #define CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG 9
- #define CFE_MISSION_SB_STATS_TLM_MSG 10
- #define CFE_MISSION_ES_APP_TLM_MSG 11
- #define CFE_MISSION_TBL_REG_TLM_MSG 12
- #define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
- #define CFE_MISSION_SB_ONESUB_TLM_MSG 14
- #define CFE_MISSION_ES_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) .

12.25.1 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

12.25.2 Macro Definition Documentation

12.25.2.1 CFE_MISSION_ES_APP_TLM_MSG

```
#define CFE_MISSION_ES_APP_TLM_MSG 11
```

Definition at line 364 of file sample_mission_cfg.h.

12.25.2.2 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 559 of file sample_mission_cfg.h.

12.25.2.3 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 243 of file sample_mission_cfg.h.

12.25.2.4 CFE_MISSION_ES_CMD_MSG

```
#define CFE_MISSION_ES_CMD_MSG 6
```

Definition at line 314 of file sample_mission_cfg.h.

12.25.2.5 CFE_MISSION_ES_CRC_16

```
#define CFE_MISSION_ES_CRC_16 2
```

CRC (16 bit additive - returns 32 bit total)

Definition at line 262 of file sample_mission_cfg.h.

Referenced by FM_ChildFileInfoCmd().

12.25.2.6 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 263 of file sample_mission_cfg.h.

Referenced by FM_ChildFileInfoCmd().

12.25.2.7 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 261 of file sample_mission_cfg.h.

Referenced by FM_ChildFileInfoCmd().

12.25.2.8 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 280 of file sample_mission_cfg.h.

12.25.2.9 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 353 of file sample_mission_cfg.h.

12.25.2.10 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 384 of file sample_mission_cfg.h.

12.25.2.11 CFE_MISSION_ES_MEMSTATS_TLM_MSG

```
#define CFE_MISSION_ES_MEMSTATS_TLM_MSG 16
```

Definition at line 368 of file sample_mission_cfg.h.

12.25.2.12 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 401 of file sample_mission_cfg.h.

12.25.2.13 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 422 of file sample_mission_cfg.h.

12.25.2.14 CFE_MISSION_ES_SEND_HK_MSG

```
#define CFE_MISSION_ES_SEND_HK_MSG 8
```

Definition at line 316 of file sample_mission_cfg.h.

12.25.2.15 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 309 of file sample_mission_cfg.h.

12.25.2.16 CFE_MISSION_EVS_HK_TLM_MSG

```
#define CFE_MISSION_EVS_HK_TLM_MSG 1
```

Definition at line 354 of file sample_mission_cfg.h.

12.25.2.17 CFE_MISSION_EVS_LONG_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_LONG_EVENT_MSG_MSG 8
```

Definition at line 361 of file sample_mission_cfg.h.

12.25.2.18 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 257 of file sample_mission_cfg.h.

12.25.2.19 CFE_MISSION_EVS_SEND_HK_MSG

```
#define CFE_MISSION_EVS_SEND_HK_MSG 9
```

Definition at line 317 of file sample_mission_cfg.h.

12.25.2.20 CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG 9
```

Definition at line 362 of file sample_mission_cfg.h.

12.25.2.21 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 539 of file sample_mission_cfg.h.

12.25.2.22 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 O↩ SAL 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 513 of file sample_mission_cfg.h.

12.25.2.23 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 486 of file sample_mission_cfg.h.

12.25.2.24 CFE_MISSION_SB_ALLSUBS_TLM_MSG

```
#define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
```

Definition at line 366 of file sample_mission_cfg.h.

12.25.2.25 CFE_MISSION_SB_CMD_MSG

```
#define CFE_MISSION_SB_CMD_MSG 3
```

Definition at line 311 of file sample_mission_cfg.h.

12.25.2.26 CFE_MISSION_SB_HK_TLM_MSG

```
#define CFE_MISSION_SB_HK_TLM_MSG 3
```

Definition at line 356 of file sample_mission_cfg.h.

12.25.2.27 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 459 of file sample_mission_cfg.h.

12.25.2.28 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.

12.25.2.29 CFE_MISSION_SB_ONESUB_TLM_MSG

```
#define CFE_MISSION_SB_ONESUB_TLM_MSG 14
```

Definition at line 367 of file sample_mission_cfg.h.

12.25.2.30 CFE_MISSION_SB_SEND_HK_MSG

```
#define CFE_MISSION_SB_SEND_HK_MSG 11
```

Definition at line 319 of file sample_mission_cfg.h.

12.25.2.31 CFE_MISSION_SB_STATS_TLM_MSG

```
#define CFE_MISSION_SB_STATS_TLM_MSG 10
```

Definition at line 363 of file sample_mission_cfg.h.

12.25.2.32 CFE_MISSION_SB_SUB_RPT_CTRL_MSG

```
#define CFE_MISSION_SB_SUB_RPT_CTRL_MSG 14
```

Definition at line 323 of file sample_mission_cfg.h.

12.25.2.33 CFE_MISSION_TBL_CMD_MSG

```
#define CFE_MISSION_TBL_CMD_MSG 4
```

Definition at line 312 of file sample_mission_cfg.h.

12.25.2.34 CFE_MISSION_TBL_HK_TLM_MSG

```
#define CFE_MISSION_TBL_HK_TLM_MSG 4
```

Definition at line 357 of file sample_mission_cfg.h.

12.25.2.35 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 442 of file sample_mission_cfg.h.

12.25.2.36 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 296 of file sample_mission_cfg.h.

12.25.2.37 CFE_MISSION_TBL_REG_TLM_MSG

```
#define CFE_MISSION_TBL_REG_TLM_MSG 12
```

Definition at line 365 of file sample_mission_cfg.h.

12.25.2.38 CFE_MISSION_TBL_SEND_HK_MSG

```
#define CFE_MISSION_TBL_SEND_HK_MSG 12
```

Definition at line 320 of file sample_mission_cfg.h.

12.25.2.39 CFE_MISSION_TEST_CMD_MSG

```
#define CFE_MISSION_TEST_CMD_MSG 2
```

Definition at line 310 of file sample_mission_cfg.h.

12.25.2.40 CFE_MISSION_TEST_HK_TLM_MSG

```
#define CFE_MISSION_TEST_HK_TLM_MSG 2
```

Definition at line 355 of file sample_mission_cfg.h.

12.25.2.41 CFE_MISSION_TIME_1HZ_CMD_MSG

```
#define CFE_MISSION_TIME_1HZ_CMD_MSG 17
```

Definition at line 326 of file sample_mission_cfg.h.

12.25.2.42 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.

12.25.2.43 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.

12.25.2.44 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.

12.25.2.45 CFE_MISSION_TIME_CFG_DEFAULT_UTC

```
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
```

Definition at line 73 of file sample_mission_cfg.h.

12.25.2.46 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.

12.25.2.47 CFE_MISSION_TIME_CMD_MSG

```
#define CFE_MISSION_TIME_CMD_MSG 5
```

Definition at line 313 of file sample_mission_cfg.h.

12.25.2.48 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 339 of file sample_mission_cfg.h.

12.25.2.49 CFE_MISSION_TIME_DEF_DELAY_SECS

```
#define CFE_MISSION_TIME_DEF_DELAY_SECS 0
```

Definition at line 167 of file sample_mission_cfg.h.

12.25.2.50 CFE_MISSION_TIME_DEF_DELAY_SUBS

```
#define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
```

Definition at line 168 of file sample_mission_cfg.h.

12.25.2.51 CFE_MISSION_TIME_DEF_LEAPS

```
#define CFE_MISSION_TIME_DEF_LEAPS 37
```

Definition at line 165 of file sample_mission_cfg.h.

12.25.2.52 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 `(time = MET + STCF - Leap Seconds)` from resulting in a negative (very large) number.

Some past missions have also created known (albeit wrong) default timestamps. For example, assume the epoch is defined as Jan 1, 1970 and further assume the default time values are set to create a timestamp of Jan 1, 2000. Even though the year 2000 timestamps are wrong, it may be of value to keep the time within some sort of bounds acceptable to the software.

Note: Sub-second units are in micro-seconds (0 to 999,999) and all values must be defined

Limits

Not Applicable

Definition at line 159 of file sample_mission_cfg.h.

12.25.2.53 CFE_MISSION_TIME_DEF_MET_SUBS

```
#define CFE_MISSION_TIME_DEF_MET_SUBS 0
```

Definition at line 160 of file sample_mission_cfg.h.

12.25.2.54 CFE_MISSION_TIME_DEF_STCF_SECS

```
#define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
```

Definition at line 162 of file sample_mission_cfg.h.

12.25.2.55 CFE_MISSION_TIME_DEF_STCF_SUBS

```
#define CFE_MISSION_TIME_DEF_STCF_SUBS 0
```

Definition at line 163 of file sample_mission_cfg.h.

12.25.2.56 CFE_MISSION_TIME_DIAG_TLM_MSG

```
#define CFE_MISSION_TIME_DIAG_TLM_MSG 6
```

Definition at line 359 of file sample_mission_cfg.h.

12.25.2.57 CFE_MISSION_TIME_EPOCH_DAY

```
#define CFE_MISSION_TIME_EPOCH_DAY 1
```

Definition at line 185 of file sample_mission_cfg.h.

12.25.2.58 CFE_MISSION_TIME_EPOCH_HOUR

```
#define CFE_MISSION_TIME_EPOCH_HOUR 0
```

Definition at line 186 of file sample_mission_cfg.h.

12.25.2.59 CFE_MISSION_TIME_EPOCH_MINUTE

```
#define CFE_MISSION_TIME_EPOCH_MINUTE 0
```

Definition at line 187 of file sample_mission_cfg.h.

12.25.2.60 CFE_MISSION_TIME_EPOCH_SECOND

```
#define CFE_MISSION_TIME_EPOCH_SECOND 0
```

Definition at line 188 of file sample_mission_cfg.h.

12.25.2.61 CFE_MISSION_TIME_EPOCH_YEAR

```
#define CFE_MISSION_TIME_EPOCH_YEAR 1980
```

Purpose Default EPOCH Values

Description:

Default ground time epoch values Note: these values are used only by the CFE_TIME_Print() API function

Limits

Year - must be within 136 years Day - Jan 1 = 1, Feb 1 = 32, etc. Hour - 0 to 23 Minute - 0 to 59 Second - 0 to 59

Definition at line 184 of file sample_mission_cfg.h.

12.25.2.62 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 226 of file sample_mission_cfg.h.

12.25.2.63 CFE_MISSION_TIME_HK_TLM_MSG

```
#define CFE_MISSION_TIME_HK_TLM_MSG 5
```

Definition at line 358 of file sample_mission_cfg.h.

12.25.2.64 CFE_MISSION_TIME_MAX_ELAPSED

```
#define CFE_MISSION_TIME_MAX_ELAPSED 200000
```

Definition at line 134 of file sample_mission_cfg.h.

12.25.2.65 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.

12.25.2.66 CFE_MISSION_TIME_SEND_CMD_MSG

```
#define CFE_MISSION_TIME_SEND_CMD_MSG 2
```

Definition at line 340 of file sample_mission_cfg.h.

12.25.2.67 CFE_MISSION_TIME_SEND_HK_MSG

```
#define CFE_MISSION_TIME_SEND_HK_MSG 13
```

Definition at line 321 of file sample_mission_cfg.h.

12.25.2.68 CFE_MISSION_TIME_TONE_CMD_MSG

```
#define CFE_MISSION_TIME_TONE_CMD_MSG 16
```

Definition at line 325 of file sample_mission_cfg.h.

12.26 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_SENDMET_PERF_ID 9

  Performance ID for Time ToneSendMET.
- #define CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID 10

  Performance ID for 1 Hz Local Task.
- #define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11

  Performance ID for 1 Hz Tone Task.

12.26.1 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:

12.26.2 Macro Definition Documentation

12.26.2.1 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.

12.26.2.2 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.

12.26.2.3 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.

12.26.2.4 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.

12.26.2.5 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.

12.26.2.6 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.

12.26.2.7 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.

12.26.2.8 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.

12.26.2.9 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.

12.26.2.10 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.

12.26.2.11 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.

12.26.2.12 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.

12.26.2.13 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.

12.27 cfe/docs/src/cfe_api.dox File Reference

12.28 cfe/docs/src/cfe_es.dox File Reference

12.29 cfe/docs/src/cfe_evs.dox File Reference

12.30 cfe/docs/src/cfe_frontpage.dox File Reference

12.31 cfe/docs/src/cfe_glossary.dox File Reference

12.32 cfe/docs/src/cfe_sb.dox File Reference

12.33 cfe/docs/src/cfe_tbl.dox File Reference

12.34 cfe/docs/src/cfe_time.dox File Reference

12.35 cfe/docs/src/cfe_xref.dox File Reference

12.36 cfe/docs/src/cfs_versions.dox File Reference

12.37 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"
```

12.37.1 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.

12.38 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.

12.38.1 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

12.38.2 Function Documentation

12.38.2.1 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

CFE_CONFIGID_UNDEFINED	if the name did not correspond to a key

12.38.2.2 CFE_Config_GetName()

```
const char* CFE_Config_GetName (
            CFE_ConfigId_t ConfigId )
```

Obtain the name of a CFE configuration ID.

Retreives 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	*ConfigId*	Configuration ID/Key to look up

Returns

Name associated with key

12.38.2.3 CFE_Config_GetObjPointer()

```
const void* CFE_Config_GetObjPointer (
            CFE_ConfigId_t ConfigId )
```

Obtain a pointer value correlating to an CFE configuration ID.

Retreives 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	*ConfigId*	Configuration ID/Key to look up

Returns

Value associated with key

Return values

NULL	if key is not defined or not set

12.38.2.4 CFE_Config_GetString()

```
const char* CFE_Config_GetString (
            CFE_ConfigId_t ConfigId )
```

Obtain a string value correlating to an CFE configuration ID.

Retreives 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	*ConfigId*	Configuration ID/Key to look up

Returns

String value associated with key

12.38.2.5 CFE_Config_GetValue()

```
uint32 CFE_Config_GetValue (
            CFE_ConfigId_t ConfigId )
```

Obtain an integer value correlating to an CFE configuration ID.

Retreives 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	*ConfigId*	Configuration ID/Key to look up

Returns

Value associated with key

Return values

0	if key is not defined or not set

12.38.2.6 CFE_Config_IterateAll()

```
void CFE_Config_IterateAll (
            void * Arg,
            CFE_Config_Callback_t Callback )
```

Iterate all known name/ID value pairs.

Parameters

in	*Arg*	User-supplied opaque argument to pass to callback
in	*Callback*	User-supplied callback function to invoke for each ID

12.39 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)

12.39.1 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

12.39.2 Macro Definition Documentation

12.39.2.1 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.

12.39.2.2 CFE_CONFIGID_UNDEFINED

```
#define CFE_CONFIGID_UNDEFINED CFE_CONFIGID_C(CFE_RESOURCEID_UNDEFINED)
```

Definition at line 49 of file cfe_config_api_typedefs.h.

12.39.3 Typedef Documentation

12.39.3.1 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.

12.39.3.2 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.

12.40 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))

12.40.1 Detailed Description

Purpose: Define macros to enforce big-endian/network byte order for 16 and 32 bit integers

12.40.2 Macro Definition Documentation

12.40.2.1 CFE_MAKE_BIG16

```
#define CFE_MAKE_BIG16(
            n ) ((((n) << 8) & 0xFF00) | (((n) >> 8) & 0x00FF))
```

Definition at line 64 of file cfe_endian.h.

12.40.2.2 CFE_MAKE_BIG32

```
#define CFE_MAKE_BIG32(
            n ) ((((n) << 24) & 0xFF000000) | (((n) << 8) & 0x00FF0000) | (((n) >> 8) & 0x0000↩
FF00) | (((n) >> 24) & 0x000000FF))
```

Definition at line 65 of file cfe_endian.h.

12.41 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)

  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)

  Max Messages Met.
- #define CFE_SB_MAX_DESTS_MET ((CFE_Status_t)0xca00000a)

  Max Destinations Met.
- #define CFE_SB_INTERNAL_ERR ((CFE_Status_t)0xca00000c)

  Internal Error.
- #define CFE_SB_WRONG_MSG_TYPE ((CFE_Status_t)0xca00000d)

  Wrong Message Type.
- #define CFE_SB_BUFFER_INVALID ((CFE_Status_t)0xca00000e)

  Buffer Invalid.
- #define CFE_SB_NOT_IMPLEMENTED ((CFE_Status_t)0xca00ffff)

  Not Implemented.
- #define CFE_TBL_ERR_INVALID_HANDLE ((CFE_Status_t)0xcc000001)

  Invalid Handle.
- #define CFE_TBL_ERR_INVALID_NAME ((CFE_Status_t)0xcc000002)

  Invalid Name.
- #define CFE_TBL_ERR_INVALID_SIZE ((CFE_Status_t)0xcc000003)

  Invalid Size.
- #define CFE_TBL_INFO_UPDATE_PENDING ((CFE_Status_t)0x4c000004)

  Update Pending.
- #define CFE_TBL_ERR_NEVER_LOADED ((CFE_Status_t)0xcc000005)

  Never Loaded.
- #define CFE_TBL_ERR_REGISTRY_FULL ((CFE_Status_t)0xcc000006)

  Registry Full.
- #define CFE_TBL_WARN_DUPLICATE ((CFE_Status_t)0x4c000007)

  Duplicate Warning.
- #define CFE_TBL_ERR_NO_ACCESS ((CFE_Status_t)0xcc000008)

  No Access.
- #define CFE_TBL_ERR_UNREGISTERED ((CFE_Status_t)0xcc000009)

  Unregistered.
- #define CFE_TBL_ERR_HANDLES_FULL ((CFE_Status_t)0xcc00000B)

  Handles Full.
- #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((CFE_Status_t)0xcc00000C)

  Duplicate Table With Different Size.
- #define CFE_TBL_ERR_DUPLICATE_NOT_OWNED ((CFE_Status_t)0xcc00000D)

  Duplicate Table And Not Owned.
- #define CFE_TBL_INFO_UPDATED ((CFE_Status_t)0x4c00000E)

  Updated.
- #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((CFE_Status_t)0xcc00000F)

  No Buffer Available.
- #define CFE_TBL_ERR_DUMP_ONLY ((CFE_Status_t)0xcc000010)

  Dump Only Error.
- #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((CFE_Status_t)0xcc000011)

  Illegal Source Type.
- #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((CFE_Status_t)0xcc000012)

  Load In Progress.

- #define CFE_TBL_ERR_FILE_TOO_LARGE ((CFE_Status_t)0xcc000014)

  File Too Large.
- #define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)

  Short File Warning.
- #define CFE_TBL_ERR_BAD_CONTENT_ID ((CFE_Status_t)0xcc000016)

  Bad Content ID.
- #define CFE_TBL_INFO_NO_UPDATE_PENDING ((CFE_Status_t)0x4c000017)

  No Update Pending.
- #define CFE_TBL_INFO_TABLE_LOCKED ((CFE_Status_t)0x4c000018)

  Table Locked.
- #define CFE_TBL_INFO_VALIDATION_PENDING ((CFE_Status_t)0x4c000019)
- #define CFE_TBL_INFO_NO_VALIDATION_PENDING ((CFE_Status_t)0x4c00001A)
- #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((CFE_Status_t)0xcc00001B)

  Bad Subtype ID.
- #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((CFE_Status_t)0xcc00001C)

  File Size Inconsistent.
- #define CFE_TBL_ERR_NO_STD_HEADER ((CFE_Status_t)0xcc00001D)

  No Standard Header.
- #define CFE_TBL_ERR_NO_TBL_HEADER ((CFE_Status_t)0xcc00001E)

  No Table Header.
- #define CFE_TBL_ERR_FILENAME_TOO_LONG ((CFE_Status_t)0xcc00001F)

  Filename Too Long.
- #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((CFE_Status_t)0xcc000020)

  File For Wrong Table.
- #define CFE_TBL_ERR_LOAD_INCOMPLETE ((CFE_Status_t)0xcc000021)

  Load Incomplete.
- #define CFE_TBL_WARN_PARTIAL_LOAD ((CFE_Status_t)0x4c000022)

  Partial Load Warning.
- #define CFE_TBL_ERR_PARTIAL_LOAD ((CFE_Status_t)0xcc000023)

  Partial Load Error.
- #define CFE_TBL_INFO_DUMP_PENDING ((CFE_Status_t)0x4c000024)

  Dump Pending.
- #define CFE_TBL_ERR_INVALID_OPTIONS ((CFE_Status_t)0xcc000025)

  Invalid Options.
- #define CFE_TBL_WARN_NOT_CRITICAL ((CFE_Status_t)0x4c000026)

  Not Critical Warning.
- #define CFE_TBL_INFO_RECOVERED_TBL ((CFE_Status_t)0x4c000027)

  Recovered Table.
- #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((CFE_Status_t)0xcc000028)

  Bad Spacecraft ID.
- #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((CFE_Status_t)0xcc000029)

  Bad Processor ID.
- #define CFE_TBL_MESSAGE_ERROR ((CFE_Status_t)0xcc00002a)

  Message Error.
- #define CFE_TBL_ERR_SHORT_FILE ((CFE_Status_t)0xcc00002b)
- #define CFE_TBL_ERR_ACCESS ((CFE_Status_t)0xcc00002c)
- #define CFE_TBL_BAD_ARGUMENT ((CFE_Status_t)0xcc00002d)

Bad Argument.

- #define CFE_TBL_NOT_IMPLEMENTED ((CFE_Status_t)0xcc00ffff)

  Not Implemented.
- #define CFE_TIME_NOT_IMPLEMENTED ((CFE_Status_t)0xce00ffff)

  Not Implemented.
- #define CFE_TIME_INTERNAL_ONLY ((CFE_Status_t)0xce000001)

  Internal Only.
- #define CFE_TIME_OUT_OF_RANGE ((CFE_Status_t)0xce000002)

  Out Of Range.
- #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((CFE_Status_t)0xce000003)

  Too Many Sync Callbacks.
- #define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000004)

  Callback Not Registered.
- #define CFE_TIME_BAD_ARGUMENT ((CFE_Status_t)0xce000005)

  Bad Argument.

Typedefs

- typedef int32 CFE_Status_t

12.41.1 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

12.41.2 Macro Definition Documentation

12.41.2.1 CFE_EVENTS_SERVICE

```
#define CFE_EVENTS_SERVICE ((CFE_Status_t)0x02000000)
```

Event Service.

Definition at line 97 of file cfe_error.h.

12.41.2.2 CFE_EXECUTIVE_SERVICE

```
#define CFE_EXECUTIVE_SERVICE ((CFE_Status_t)0x04000000)
```

Executive Service.

Definition at line 98 of file cfe_error.h.

12.41.2.3 CFE_FILE_SERVICE

```
#define CFE_FILE_SERVICE ((CFE_Status_t)0x06000000)
```

File Service.

Definition at line 99 of file cfe_error.h.

12.41.2.4 CFE_GENERIC_SERVICE

```
#define CFE_GENERIC_SERVICE ((CFE_Status_t)0x08000000)
```

Generic Service.

Definition at line 100 of file cfe_error.h.

12.41.2.5 CFE_SERVICE_BITMASK

```
#define CFE_SERVICE_BITMASK ((CFE_Status_t)0x0e000000)
```

Error Service Bitmask.

Definition at line 95 of file cfe_error.h.

12.41.2.6 CFE_SEVERITY_BITMASK

```
#define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)
```

Error Severity Bitmask.

Definition at line 86 of file cfe_error.h.

12.41.2.7 CFE_SEVERITY_ERROR

```
#define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)
```

Severity Error.

Definition at line 90 of file cfe_error.h.

12.41.2.8 CFE_SEVERITY_INFO

```
#define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)
```

Severity Info.

Definition at line 89 of file cfe_error.h.

12.41.2.9 CFE_SEVERITY_SUCCESS

```
#define CFE_SEVERITY_SUCCESS ((CFE_Status_t)0x00000000)
```

Severity Success.

Definition at line 88 of file cfe_error.h.

12.41.2.10 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.

12.41.2.11 CFE_TABLE_SERVICE

```
#define CFE_TABLE_SERVICE ((CFE_Status_t)0x0c000000)
```

Table Service.

Definition at line 102 of file cfe_error.h.

12.41.2.12 CFE_TIME_SERVICE

```
#define CFE_TIME_SERVICE ((CFE_Status_t)0x0e000000)
```

Time Service.

Definition at line 103 of file cfe_error.h.

12.41.3 Typedef Documentation

12.41.3.1 CFE_Status_t

```
typedef int32 CFE_Status_t
```

Definition at line 43 of file cfe_error.h.

12.42 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 ModeId, 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)

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_ResourceId_t ResourceId)

  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_Task↩Priority_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 Buffer↩Length)

  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.

12.42.1 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:

12.42.2 Macro Definition Documentation

12.42.2.1 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.

12.42.2.2 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.

12.42.2.3 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.

12.42.2.4 OS_PRINTF

```
#define OS_PRINTF(
              m,
              n )
```

Definition at line 50 of file cfe_es.h.

12.43 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_UNDEFIN↩ED)
- #define CFE_ES_MEMHANDLE_UNDEFINED CFE_ES_MEMHANDLE_C(CFE_RESOURCEID_UNDEFI↩NED)
- #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.

12.43.1 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:

12.43.2 Macro Definition Documentation

12.43.2.1 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.

12.43.2.2 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.

12.43.2.3 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.

12.43.2.4 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.

12.43.2.5 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.

12.43.2.6 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.

12.43.2.7 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.

12.43.2.8 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.

12.43.2.9 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.

12.43.2.10 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.

12.43.2.11 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.

12.43.2.12 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.

12.43.2.13 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.

12.43.2.14 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.

12.43.2.15 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.

12.43.2.16 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.

12.43.2.17 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.

12.43.2.18 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.

12.43.3 Typedef Documentation

12.43.3.1 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.

12.43.3.2 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.

12.43.3.3 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.

12.43.3.4 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.

12.43.3.5 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.

12.43.3.6 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.

12.44 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_App↩Type_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_C↩ORE_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.

12.44.1 Detailed Description

Declarations and prototypes for cfe_es_extern_typedefs module

12.44.2 Macro Definition Documentation

12.44.2.1 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.

12.44.2.2 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.

12.44.3 Typedef Documentation

12.44.3.1 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.

12.44.3.2 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.

12.44.3.3 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.

12.44.3.4 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.

12.44.3.5 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.

12.44.3.6 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.

12.44.3.7 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.

12.44.3.8 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.

12.44.3.9 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.

12.44.3.10 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.

12.44.3.11 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.

12.44.3.12 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.

12.44.3.13 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.

12.44.3.14 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.

12.44.3.15 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.

12.44.3.16 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

12.44.3.17 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.

12.44.3.18 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.

12.44.3.19 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.

12.44.3.20 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.

12.44.3.21 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.

12.44.4 Enumeration Type Documentation

12.44.4.1 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.

12.44.4.2 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.

12.44.4.3 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.

12.44.4.4 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.

12.44.4.5 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.

12.44.4.6 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.

12.44.4.7 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.

12.45 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.

12.45.1 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

12.45.2 Macro Definition Documentation

12.45.2.1 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.

12.45.2.2 CFE_EVS_SendCrit

```
#define CFE_EVS_SendCrit(
            E,
            ...  ) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)
```

Definition at line 50 of file cfe_evs.h.

12.45.2.3 CFE_EVS_SendDbg

```
#define CFE_EVS_SendDbg(
            E,
            ...  ) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)
```

Definition at line 47 of file cfe_evs.h.

12.45.2.4 CFE_EVS_SendErr

```
#define CFE_EVS_SendErr(
            E,
            ...  ) CFE_EVS_Send(E, ERROR, __VA_ARGS__)
```

Definition at line 49 of file cfe_evs.h.

12.45.2.5 CFE_EVS_SendInfo

```
#define CFE_EVS_SendInfo(
            E,
            ...  ) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)
```

Definition at line 48 of file cfe_evs.h.

12.46 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 0xFFFC

  Sends the first 4 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_8_STOP 0xFFF8

  Sends the first 8 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_16_STOP 0xFFF0

  Sends the first 16 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_32_STOP 0xFFE0

  Sends the first 32 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_64_STOP 0xFFC0

  Sends the first 64 events. All remaining messages are filtered.
- #define CFE_EVS_EVERY_OTHER_ONE 0x0001

  Sends every other event.
- #define CFE_EVS_EVERY_OTHER_TWO 0x0002

  Sends two, filters one, sends two, filters one, etc.
- #define CFE_EVS_EVERY_FOURTH_ONE 0x0003

  Sends every fourth event message. All others are filtered.

Typedefs

- typedef struct CFE_EVS_BinFilter CFE_EVS_BinFilter_t

  Event message filter definition structure.

12.46.1 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

12.46.2 Macro Definition Documentation

12.46.2.1 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.

12.46.2.2 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.

12.46.2.3 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.

12.46.2.4 CFE_EVS_FIRST_16_STOP

```
#define CFE_EVS_FIRST_16_STOP 0xFFF0
```

Sends the first 16 events. All remaining messages are filtered.

Definition at line 49 of file cfe_evs_api_typedefs.h.

12.46.2.5 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.

12.46.2.6 CFE_EVS_FIRST_4_STOP

```
#define CFE_EVS_FIRST_4_STOP 0xFFFC
```

Sends the first 4 events. All remaining messages are filtered.

Definition at line 47 of file cfe_evs_api_typedefs.h.

12.46.2.7 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.

12.46.2.8 CFE_EVS_FIRST_8_STOP

```
#define CFE_EVS_FIRST_8_STOP 0xFFF8
```

Sends the first 8 events. All remaining messages are filtered.

Definition at line 48 of file cfe_evs_api_typedefs.h.

12.46.2.9 CFE_EVS_FIRST_ONE_STOP

```
#define CFE_EVS_FIRST_ONE_STOP 0xFFFF
```

Sends the first event. All remaining messages are filtered.

Definition at line 45 of file cfe_evs_api_typedefs.h.

12.46.2.10 CFE_EVS_FIRST_TWO_STOP

```
#define CFE_EVS_FIRST_TWO_STOP 0xFFFE
```

Sends the first 2 events. All remaining messages are filtered.

Definition at line 46 of file cfe_evs_api_typedefs.h.

12.46.2.11 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.

12.46.3 Typedef Documentation

12.46.3.1 CFE_EVS_BinFilter_t

```
typedef struct CFE_EVS_BinFilter CFE_EVS_BinFilter_t
```

Event message filter definition structure.

12.47 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.

12.47.1 Detailed Description

Declarations and prototypes for cfe_evs_extern_typedefs module

12.47.2 Typedef Documentation

12.47.2.1 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.

12.47.2.2 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.

12.47.2.3 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.

12.47.2.4 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.

12.47.2.5 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.

12.47.3 Enumeration Type Documentation

12.47.3.1 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.

12.47.3.2 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.

12.47.3.3 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.

12.47.3.4 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.

12.47.3.5 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.

12.48 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference

```
#include "common_types.h"
#include "osconfig.h"
#include "cfe_platform_cfg.h"
#include "cfe_error.h"
#include "cfe_fs_api_typedefs.h"
#include "cfe_fs_extern_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

Functions

- CFE_Status_t CFE_FS_ReadHeader (CFE_FS_Header_t ∗Hdr, osal_id_t FileDes)

  Read the contents of the Standard cFE File Header.
- void CFE_FS_InitHeader (CFE_FS_Header_t ∗Hdr, const char ∗Description, uint32 SubType)

  Initializes the contents of the Standard cFE File Header.
- CFE_Status_t CFE_FS_WriteHeader (osal_id_t FileDes, CFE_FS_Header_t ∗Hdr)

  Write the specified Standard cFE File Header to the specified file.
- CFE_Status_t CFE_FS_SetTimestamp (osal_id_t FileDes, CFE_TIME_SysTime_t NewTimestamp)

  Modifies the Time Stamp field in the Standard cFE File Header for the specified file.
- const char ∗ CFE_FS_GetDefaultMountPoint (CFE_FS_FileCategory_t FileCategory)

  Get the default virtual mount point for a file category.
- const char ∗ CFE_FS_GetDefaultExtension (CFE_FS_FileCategory_t FileCategory)

  Get the default filename extension for a file category.
- int32 CFE_FS_ParseInputFileNameEx (char ∗OutputBuffer, const char ∗InputBuffer, size_t OutputBufSize, size↩_t InputBufSize, const char ∗DefaultInput, const char ∗DefaultPath, const char ∗DefaultExtension)

  Parse a filename input from an input buffer into a local buffer.
- int32 CFE_FS_ParseInputFileName (char ∗OutputBuffer, const char ∗InputName, size_t OutputBufSize, CFE_↩FS_FileCategory_t FileCategory)

  Parse a filename string from the user into a local buffer.
- CFE_Status_t CFE_FS_ExtractFilenameFromPath (const char ∗OriginalPath, char ∗FileNameOnly)

  Extracts the filename from a unix style path and filename string.
- int32 CFE_FS_BackgroundFileDumpRequest (CFE_FS_FileWriteMetaData_t ∗Meta)

  Register a background file dump request.
- bool CFE_FS_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t ∗Meta)

  Query if a background file write request is currently pending.

12.48.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.49 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_UNKNOWN, CFE_FS_FileCategory_DYNAMIC_MODULE, CFE_FS_FileCategory_BI↩NARY_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_CR↩EATE_ERROR, CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR, CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR, CFE_FS_FileWriteEvent_MAX }

12.49.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

12.49.2 Typedef Documentation

12.49.2.1 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

in,out	*Meta*	Pointer to the metadata object
in	*RecordNum*	Incrementing record number counter
out	*Buffer*	Pointer to buffer data block, should be set by implementation
out	*BufSize*	Pointer to buffer data size, should be set by implementation

Returns

End of file status

Return values

true	if at last data record, and output file should be closed
false	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.

12.49.2.2 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).

12.49.2.3 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	*Meta*	Pointer to the metadata object
in	*Event*	Generalized type of event to report (not actual event ID)
in	*Status*	Generalized status code (may be from OSAL or CFE)
in	*RecordNum*	Record number counter at which event occurred
in	*BlockSize*	Size of record being processed when event occurred (if applicable)
in	*Position*	File position/size when event occurred

Definition at line 114 of file cfe_fs_api_typedefs.h.

12.49.3 Enumeration Type Documentation

12.49.3.1 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.

12.49.3.2 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.

12.50 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_PIPEDATA = 20, CFE_FS_SubType_SB_↩ROUTEDATA = 21, CFE_FS_SubType_SB_MAPDATA = 22,
  CFE_FS_SubType_ES_QUERYALLTASKS = 23 }

  File subtypes used within cFE.

12.50.1 Detailed Description

Declarations and prototypes for cfe_fs_extern_typedefs module

12.50.2 Macro Definition Documentation

12.50.2.1 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.

12.50.2.2 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.

12.50.3 Typedef Documentation

12.50.3.1 CFE_FS_Header_t

```
typedef struct CFE_FS_Header CFE_FS_Header_t
```

Standard cFE File header structure definition.

12.50.3.2 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.

12.50.4 Enumeration Type Documentation

12.50.4.1 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_PIPEDATA	Software Bus Pipe Data Dump File. Software Bus Pipe Data Dump File which is generated in response to a $sc_$cpu_SB_WritePipe2File command.
CFE_FS_SubType_SB_ROUTEDATA	Software Bus Message Routing Data Dump File. Software Bus Message Routing Data Dump File which is generated in response to a $sc_$cpu_SB_WriteRouting2File command.
CFE_FS_SubType_SB_MAPDATA	Software Bus Message Mapping Data Dump File. Software Bus Message Mapping Data Dump File which is generated in response to a $sc_$cpu_SB_WriteMap2File command.
CFE_FS_SubType_ES_QUERYALLTASKS	Executive Services Query All Tasks Data File. Executive Services Query All Tasks Data File which is generated in response to a $sc_$cpu_ES_WriteTaskInfo2File command.

Definition at line 61 of file cfe_fs_extern_typedefs.h.

12.51 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_msg_hdr.h"
#include "cfe_msg_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

Functions

- CFE_Status_t CFE_MSG_Init (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId, CFE_MSG_Size_t Size)

  Initialize a message.
- CFE_Status_t CFE_MSG_GetSize (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t *Size)

  Gets the total size of a message.
- CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)

  Sets the total size of a message.
- CFE_Status_t CFE_MSG_GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)

  Gets the message type.
- CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)

  Sets the message type.

- CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)

  Gets the message header version.
- CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)

  Sets the message header version.
- CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)

  Gets the message secondary header boolean.
- CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)

  Sets the message secondary header boolean.
- CFE_Status_t CFE_MSG_GetApId (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t *ApId)

  Gets the message application ID.
- CFE_Status_t CFE_MSG_SetApId (CFE_MSG_Message_t *MsgPtr, CFE_MSG_ApId_t ApId)

  Sets the message application ID.
- CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)

  Gets the message segmentation flag.
- CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)

  Sets the message segmentation flag.
- CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)

  Gets the message sequence count.
- CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)

  Sets the message sequence count.
- CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)

  Gets the next sequence count value (rolls over if appropriate)
- CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)

  Gets the message EDS version.
- CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)

  Sets the message EDS version.
- CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)

  Gets the message endian.
- CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)

  Sets the message endian.
- CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)

  Gets the message playback flag.
- CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)

  Sets the message playback flag.
- CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)

  Gets the message subsystem.
- CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)

Sets the message subsystem.

- CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)

  Gets the message system.
- CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)

  Sets the message system.
- CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t *MsgPtr)

  Calculates and sets the checksum of a message.
- CFE_Status_t CFE_MSG_ValidateChecksum (const CFE_MSG_Message_t *MsgPtr, bool *IsValid)

  Validates the checksum of a message.
- CFE_Status_t CFE_MSG_SetFcnCode (CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode)

  Sets the function code field in a message.
- CFE_Status_t CFE_MSG_GetFcnCode (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *Fcn↩Code)

  Gets the function code field from a message.
- CFE_Status_t CFE_MSG_GetMsgTime (const CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t *Time)

  Gets the time field from a message.
- CFE_Status_t CFE_MSG_SetMsgTime (CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t NewTime)

  Sets the time field in a message.
- CFE_Status_t CFE_MSG_GetMsgId (const CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t *MsgId)

  Gets the message id from a message.
- CFE_Status_t CFE_MSG_SetMsgId (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId)

  Sets the message id bits in a message.
- CFE_Status_t CFE_MSG_GetTypeFromMsgId (CFE_SB_MsgId_t MsgId, CFE_MSG_Type_t *Type)

  Gets message type using message ID.

12.51.1 Detailed Description

Message access APIs

12.52 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_ApId_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.

12.52.1 Detailed Description

Typedefs for Message API

- Separate from API so these can be adjusted for custom implementations

12.52.2 Macro Definition Documentation

12.52.2.1 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.

12.52.2.2 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.

12.52.2.3 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.

12.52.3 Typedef Documentation

12.52.3.1 CFE_MSG_ApId_t

```
typedef uint16 CFE_MSG_ApId_t
```

Message application ID.

Definition at line 50 of file cfe_msg_api_typedefs.h.

12.52.3.2 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.

12.52.3.3 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.

12.52.3.4 CFE_MSG_EDSVersion_t

```
typedef uint16 CFE_MSG_EDSVersion_t
```

Message EDS version.

Definition at line 52 of file cfe_msg_api_typedefs.h.

12.52.3.5 CFE_MSG_Endian_t

```
typedef enum CFE_MSG_Endian CFE_MSG_Endian_t
```

Endian flag.

12.52.3.6 CFE_MSG_FcnCode_t

```
typedef uint16 CFE_MSG_FcnCode_t
```

Message function code.

Definition at line 48 of file cfe_msg_api_typedefs.h.

12.52.3.7 CFE_MSG_HeaderVersion_t

```
typedef uint16 CFE_MSG_HeaderVersion_t
```

Message header version.

Definition at line 49 of file cfe_msg_api_typedefs.h.

12.52.3.8 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.

12.52.3.9 CFE_MSG_PlaybackFlag_t

```
typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t
```

Playback flag.

12.52.3.10 CFE_MSG_SegmentationFlag_t

```
typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t
```

Segmentation flags.

12.52.3.11 CFE_MSG_SequenceCount_t

```
typedef uint16 CFE_MSG_SequenceCount_t
```

Message sequence count.

Definition at line 51 of file cfe_msg_api_typedefs.h.

12.52.3.12 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.

12.52.3.13 CFE_MSG_Subsystem_t

```
typedef uint16 CFE_MSG_Subsystem_t
```

Message subsystem.

Definition at line 53 of file cfe_msg_api_typedefs.h.

12.52.3.14 CFE_MSG_System_t

```
typedef uint16 CFE_MSG_System_t
```

Message system.

Definition at line 54 of file cfe_msg_api_typedefs.h.

12.52.3.15 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.

12.52.3.16 CFE_MSG_Type_t

```
typedef enum CFE_MSG_Type CFE_MSG_Type_t
```

Message type.

12.52.4 Enumeration Type Documentation

12.52.4.1 CFE_MSG_Endian

```
enum CFE_MSG_Endian
```

Endian flag.

Enumerator

CFE_MSG_Endian_Invalid	Invalid endian setting.
CFE_MSG_Endian_Big	Big endian.
CFE_MSG_Endian_Little	Little endian.

Definition at line 75 of file cfe_msg_api_typedefs.h.

12.52.4.2 CFE_MSG_PlaybackFlag

`enum CFE_MSG_PlaybackFlag`

Playback flag.

Enumerator

CFE_MSG_PlayFlag_Invalid	Invalid playback setting.
CFE_MSG_PlayFlag_Original	Original.
CFE_MSG_PlayFlag_Playback	Playback.

Definition at line 83 of file cfe_msg_api_typedefs.h.

12.52.4.3 CFE_MSG_SegmentationFlag

`enum CFE_MSG_SegmentationFlag`

Segmentation flags.

Enumerator

CFE_MSG_SegFlag_Invalid	Invalid segmentation flag.
CFE_MSG_SegFlag_Continue	Continuation segment of User Data.
CFE_MSG_SegFlag_First	First segment of User Data.
CFE_MSG_SegFlag_Last	Last segment of User Data.
CFE_MSG_SegFlag_Unsegmented	Unsegmented data.

Definition at line 65 of file cfe_msg_api_typedefs.h.

12.52.4.4 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.

12.53 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_UNWR↩AP(id))

  Determine if a derived (app-specific) ID is defined or not.
- #define CFE_RESOURCEID_TEST_EQUAL(id1, id2) CFE_ResourceId_Equal(CFE_RESOURCEID_UNWR↩AP(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.

12.53.1 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)

12.53.2 Macro Definition Documentation

12.53.2.1 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.

12.53.2.2 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.

12.53.2.3 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_ResourceId_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.

12.53.3 Function Documentation

12.53.3.1 CFE_ResourceId_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	*id1*	Resource ID to check
in	*id2*	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_ResourceId_IsDefined().

12.53.3.2 CFE_ResourceId_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	*StartId*	the last issued ID for the resource category (app, lib, etc).
in	*TableSize*	the maximum size of the target table
in	*CheckFunc*	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_ResourceId_IsDefined().

12.53.3.3 CFE_ResourceId_FromInteger()

```
static CFE_ResourceId_t CFE_ResourceId_FromInteger (
            unsigned long Value )  [inline], [static]
```

Convert an integer to a resource ID.

This is the inverse of CFE_ResourceId_ToInteger(), and reconstitutes the original CFE_ResourceId_t value from the integer representation.

This may be used, for instance, where an ID value is parsed from a text file or message using C library APIs such as scanf() or strtoul().

See also

CFE_ResourceId_ToInteger()

Parameters

in	*Value*	Integer value to convert

Returns

ID value corresponding to integer

Definition at line 121 of file cfe_resourceid.h.

12.53.3.4 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().

12.53.3.5 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	*Resource↩ Id*	the resource ID to decode

Returns

The serial number associated with that ID

Referenced by CFE_ResourceId_IsDefined().

12.53.3.6 CFE_ResourceId_IsDefined()

```
static bool CFE_ResourceId_IsDefined (
            CFE_ResourceId_t id )  [inline], [static]
```

Check if a resource ID value is defined.

The constant CFE_RESOURCEID_UNDEFINED represents an undefined ID value, such that the expression:

```
CFE_ResourceId_IsDefined(CFE_RESOURCEID_UNDEFINED)
```

Always returns false.

Parameters

in	*id*	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_ResourceId_Equal(), CFE_ResourceId_FindNext(), CFE_ResourceId_GetBase(), CFE_Resource↩ Id_GetSerial(), CFE_ResourceId_ToIndex(), and CFE_RESOURCEID_UNDEFINED.

Here is the call graph for this function:

12.53.3.7 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	*Id*	The resource ID
in	*BaseValue*	The respective ID base value corresponding to the ID type
in	*TableSize*	The actual size of the internal table (MAX index value + 1)
out	*Idx*	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_ResourceId_IsDefined().

12.53.3.8 CFE_ResourceId_ToInteger()

```
static unsigned long CFE_ResourceId_ToInteger (
            CFE_ResourceId_t id )  [inline], [static]
```

Convert a resource ID to an integer.

This is primarily intended for logging purposes, such was writing to debug console, event messages, or log files, using printf-like APIs.

For compatibility with C library APIs, this returns an "unsigned long" type and should be used with the "%lx" format specifier in a printf format string.

Note

No assumptions should be made about the actual integer value, such as its base/range. It may be printed, but should not be modified or tested/compared using other arithmetic ops, and should never be used as the index to an array or table. See the related function CFE_ResourceId_ToIndex() for cases where a zero-based array/table index is needed.

See also

CFE_ResourceId_FromInteger()

Parameters

in	*id*	Resource ID to convert

Returns

Integer value corresponding to ID

Definition at line 102 of file cfe_resourceid.h.

12.54 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.

12.54.1 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)

12.54.2 Macro Definition Documentation

12.54.2.1 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.

12.54.2.2 CFE_RESOURCEID_UNDEFINED

```
#define CFE_RESOURCEID_UNDEFINED ((CFE_ResourceId_t)CFE_RESOURCEID_WRAP(0))
```

A resource ID value that represents an undefined/unused resource.

This constant may be used to initialize local variables of the CFE_ResourceId_t type to a safe value that will not alias a valid ID.

By design, this value is also the result of zeroing a CFE_ResourceId_t type via standard functions like memset(), such that objects initialized using this method will also be set to safe values.

Definition at line 65 of file cfe_resourceid_api_typedefs.h.

Referenced by CFE_ResourceId_IsDefined().

12.55 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_PipeId_t *PipeIdPtr, uint16 Depth, const char *PipeName)

  Creates a new software bus pipe.
- CFE_Status_t CFE_SB_DeletePipe (CFE_SB_PipeId_t PipeId)

  Delete a software bus pipe.
- CFE_Status_t CFE_SB_PipeId_ToIndex (CFE_SB_PipeId_t PipeID, uint32 *Idx)

  Obtain an index value correlating to an SB Pipe ID.
- CFE_Status_t CFE_SB_SetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 Opts)

  Set options on a pipe.
- CFE_Status_t CFE_SB_GetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 *OptsPtr)

  Get options on a pipe.
- CFE_Status_t CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_PipeId_t PipeId)

  Get the pipe name for a given id.
- CFE_Status_t CFE_SB_GetPipeIdByName (CFE_SB_PipeId_t *PipeIdPtr, const char *PipeName)

  Get pipe id by pipe name.
- 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.

- 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.
- 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.

12.55.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.

Author: R.McGraw/SSI

12.55.2 Macro Definition Documentation

12.55.2.1 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.

12.55.2.2 CFE_CLR

```
#define CFE_CLR(
            i,
            x ) ((i) &= ~CFE_BIT(x))
```

Clears bit x of i.

Definition at line 46 of file cfe_sb.h.

12.55.2.3 CFE_SET

```
#define CFE_SET(
            i,
            x ) ((i) |= CFE_BIT(x))
```

Sets bit x of i.

Definition at line 45 of file cfe_sb.h.

12.55.2.4 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.

12.56 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_PipeId_t)CFE_RESOURCEID_WRAP(val))

  Cast/Convert a generic CFE_ResourceId_t to a CFE_SB_PipeId_t.
- #define CFE_SB_INVALID_PIPE CFE_SB_PIPEID_C(CFE_RESOURCEID_UNDEFINED)

  A CFE_SB_PipeId_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.

12.56.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.

Author: R.McGraw/SSI

12.56.2 Macro Definition Documentation

12.56.2.1 CFE_SB_DEFAULT_QOS

```
#define CFE_SB_DEFAULT_QOS ((CFE_SB_Qos_t) {0})
```

Default Qos macro.

Definition at line 135 of file cfe_sb_api_typedefs.h.

12.56.2.2 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 FM_ProcessPkt().

12.56.2.3 CFE_SB_INVALID_PIPE

```
#define CFE_SB_INVALID_PIPE CFE_SB_PIPEID_C(CFE_RESOURCEID_UNDEFINED)
```

A CFE_SB_PipeId_t value which is always invalid.

This may be used as a safe initializer for CFE_SB_PipeId_t values

Definition at line 125 of file cfe_sb_api_typedefs.h.

12.56.2.4 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().

12.56.2.5 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.

12.56.2.6 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().

12.56.2.7 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.

12.56.2.8 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.

12.56.2.9 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.

12.56.2.10 CFE_SB_POLL

```
#define CFE_SB_POLL 0
```

Option used with CFE_SB_ReceiveBuffer to request immediate pipe status.

Definition at line 45 of file cfe_sb_api_typedefs.h.

12.56.2.11 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.

12.56.2.12 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.

12.56.3 Typedef Documentation

12.56.3.1 CFE_SB_Buffer_t

```
typedef union CFE_SB_Msg CFE_SB_Buffer_t
```

Software Bus generic message.

12.57 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_RouteId_Atom_t

  An integer type that should be used for indexing into the Routing Table.
- typedef uint32 CFE_SB_MsgId_Atom_t

  CFE_SB_MsgId_Atom_t primitive type definition.
- typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t

  CFE_SB_PipeId_t to primitive type definition.

Enumerations

- enum CFE_SB_QosPriority { CFE_SB_QosPriority_LOW = 0, CFE_SB_QosPriority_HIGH = 1 }

  Label definitions associated with CFE_SB_QosPriority_Enum_t.
- enum CFE_SB_QosReliability { CFE_SB_QosReliability_LOW = 0, CFE_SB_QosReliability_HIGH = 1 }

  Label definitions associated with CFE_SB_QosReliability_Enum_t.

12.57.1 Detailed Description

Declarations and prototypes for cfe_sb_extern_typedefs module

12.57.2 Macro Definition Documentation

12.57.2.1 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.

12.57.3 Typedef Documentation

12.57.3.1 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.

12.57.3.2 CFE_SB_PipeId_t

```
typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t
```

CFE_SB_PipeId_t to primitive type definition.

Software Bus pipe identifier used in many SB APIs, as well as SB Telemetry messages and data files.

Definition at line 126 of file cfe_sb_extern_typedefs.h.

12.57.3.3 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.

12.57.3.4 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.

12.57.3.5 CFE_SB_RouteId_Atom_t

```
typedef uint16 CFE_SB_RouteId_Atom_t
```

An integer type that should be used for indexing into the Routing Table.

Definition at line 95 of file cfe_sb_extern_typedefs.h.

12.57.4 Enumeration Type Documentation

12.57.4.1 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.

12.57.4.2 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.

12.58 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 Tbl↩Handles[ ])

  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.

12.58.1 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:

12.59 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_time_extern_typedefs.h"
```

Data Structures

- struct CFE_TBL_Info

  Table Info.

Macros

- #define CFE_TBL_OPT_BUFFER_MSK (0x0001)

  Table buffer mask.
- #define CFE_TBL_OPT_SNGL_BUFFER (0x0000)

  Single buffer table.
- #define CFE_TBL_OPT_DBL_BUFFER (0x0001)

  Double buffer table.
- #define CFE_TBL_OPT_LD_DMP_MSK (0x0002)

  Table load/dump mask.
- #define CFE_TBL_OPT_LOAD_DUMP (0x0000)

  Load/Dump table.
- #define CFE_TBL_OPT_DUMP_ONLY (0x0002)

Dump only table.

- #define CFE_TBL_OPT_USR_DEF_MSK (0x0004)

  Table user defined mask.
- #define CFE_TBL_OPT_NOT_USR_DEF (0x0000)

  Not user defined table.
- #define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)

  User Defined table,.
- #define CFE_TBL_OPT_CRITICAL_MSK (0x0008)

  Table critical mask.
- #define CFE_TBL_OPT_NOT_CRITICAL (0x0000)

  Not critical table.
- #define CFE_TBL_OPT_CRITICAL (0x0008)

  Critical table.
- #define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)

  Default table options.
- #define CFE_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_FULL_NAME_LEN)

  Table maximum full name length.
- #define CFE_TBL_BAD_TABLE_HANDLE (CFE_TBL_Handle_t)0xFFFF

  Bad table handle.

Typedefs

- typedef int32(∗ CFE_TBL_CallbackFuncPtr_t) (void ∗TblPtr)

  Table Callback Function.
- typedef int16 CFE_TBL_Handle_t

  Table Handle primitive.
- typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t

  Table Source.
- typedef struct CFE_TBL_Info CFE_TBL_Info_t

  Table Info.

Enumerations

- enum CFE_TBL_SrcEnum { CFE_TBL_SRC_FILE = 0, CFE_TBL_SRC_ADDRESS }

  Table Source.

12.59.1 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:

12.59.2 Macro Definition Documentation

12.59.2.1 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.

12.59.2.2 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.

12.59.3 Typedef Documentation

12.59.3.1 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.

12.59.3.2 CFE_TBL_Handle_t

```
typedef int16 CFE_TBL_Handle_t
```

Table Handle primitive.

Definition at line 87 of file cfe_tbl_api_typedefs.h.

12.59.3.3 CFE_TBL_Info_t

```
typedef struct CFE_TBL_Info CFE_TBL_Info_t
```

Table Info.

12.59.3.4 CFE_TBL_SrcEnum_t

```
typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t
```

Table Source.

12.59.4 Enumeration Type Documentation

12.59.4.1 CFE_TBL_SrcEnum

```
enum CFE_TBL_SrcEnum
```

Table Source.

Enumerator

CFE_TBL_SRC_FILE	File source When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table.
CFE_TBL_SRC_ADDRESS	Address source When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the CFE_TBL_Register function Size parameter.

Definition at line 90 of file cfe_tbl_api_typedefs.h.

12.60 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.

12.60.1 Detailed Description

Declarations and prototypes for cfe_tbl_extern_typedefs module

12.60.2 Typedef Documentation

12.60.2.1 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_typedefs.h.

12.60.2.2 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.

12.60.3 Enumeration Type Documentation

12.60.3.1 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.

12.61 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

12.61.1 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:

12.61.2 Macro Definition Documentation

12.61.2.1 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.

12.61.3 Typedef Documentation

12.61.3.1 CFE_TBL_FileDef_t

```
typedef struct CFE_TBL_FileDef CFE_TBL_FileDef_t
```

12.62 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.

12.62.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.62.2 Macro Definition Documentation

12.62.2.1 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.

12.63 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.

12.63.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

12.63.2 Macro Definition Documentation

12.63.2.1 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.

12.63.3 Typedef Documentation

12.63.3.1 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.

12.63.3.2 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.

12.63.4 Enumeration Type Documentation

12.63.4.1 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.

12.64 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.

12.64.1 Detailed Description

Declarations and prototypes for cfe_time_extern_typedefs module

12.64.2 Typedef Documentation

12.64.2.1 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.

12.64.2.2 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.

12.64.2.3 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.

12.64.2.4 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.

12.64.2.5 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.

12.64.2.6 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.

12.64.2.7 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.

12.64.2.8 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.

12.64.3 Enumeration Type Documentation

12.64.3.1 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.

12.64.3.2 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.

12.64.3.3 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.

12.64.3.4 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.

12.64.3.5 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.

12.64.3.6 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.

12.64.3.7 CFE_TIME_ToneSignalSelect

`enum CFE_TIME_ToneSignalSelect`

Label definitions associated with CFE_TIME_ToneSignalSelect_Enum_t.

Enumerator

CFE_TIME_ToneSignalSelect_PRIMARY	Primary Source.
CFE_TIME_ToneSignalSelect_REDUNDANT	Redundant Source.

Definition at line 206 of file cfe_time_extern_typedefs.h.

12.65 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference

Macros

- #define CFE_BUILD_NUMBER 110

  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.

12.65.1 Detailed Description

Provide version identifiers for the cFE core. See Version Numbers for further details.

12.65.2 Macro Definition Documentation

12.65.2.1 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.

12.65.2.2 CFE_BUILD_NUMBER

```
#define CFE_BUILD_NUMBER 110
```

Development: Number of development git commits since CFE_BUILD_BASELINE.

Definition at line 29 of file cfe_version.h.

12.65.2.3 CFE_MAJOR_VERSION

```
#define CFE_MAJOR_VERSION 6
```

Major version number.

Definition at line 33 of file cfe_version.h.

12.65.2.4 CFE_MINOR_VERSION

```
#define CFE_MINOR_VERSION 7
```

Minor version number.

Definition at line 34 of file cfe_version.h.

12.65.2.5 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.

12.65.2.6 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.

12.65.2.7 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.

12.65.2.8 CFE_STR

```
#define CFE_STR(
              x ) CFE_STR_HELPER(x)
```

Expand macro before conversion.

Definition at line 47 of file cfe_version.h.

12.65.2.9 CFE_STR_HELPER

```
#define CFE_STR_HELPER(
              x ) #x
```

Convert argument to string.

Definition at line 46 of file cfe_version.h.

12.65.2.10 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.

12.66 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

ES Start Performance Analyzer Data Collection Command Success Event ID.

- #define CFE_ES_PERF_STARTCMD_ERR_EID 58

  ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.
- #define CFE_ES_PERF_STARTCMD_TRIG_ERR_EID 59

  ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.
- #define CFE_ES_PERF_STOPCMD_EID 60

  ES Stop Performance Analyzer Data Collection Command Request Success Event ID.
- #define CFE_ES_PERF_STOPCMD_ERR2_EID 62

  ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.
- #define CFE_ES_PERF_FILTMSKCMD_EID 63

  ES Set Performance Analyzer Filter Mask Command Success Event ID.
- #define CFE_ES_PERF_FILTMSKERR_EID 64

  ES Set Performance Analyzer Filter Mask Command Invalid Index Event ID.
- #define CFE_ES_PERF_TRIGMSKCMD_EID 65

  ES Set Performance Analyzer Trigger Mask Command Success Event ID.
- #define CFE_ES_PERF_TRIGMSKERR_EID 66

  ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.
- #define CFE_ES_PERF_LOG_ERR_EID 67

  ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.
- #define CFE_ES_PERF_DATAWRITTEN_EID 68

  Performance Log Write Success Event ID.
- #define CFE_ES_CDS_REGISTER_ERR_EID 69

  ES Register CDS API Failed Event ID.
- #define CFE_ES_SYSLOGMODE_EID 70

  ES Set System Log Overwrite Mode Command Success Event ID.
- #define CFE_ES_ERR_SYSLOGMODE_EID 71

  ES Set System Log Overwrite Mode Command Failed Event ID.
- #define CFE_ES_RESET_PR_COUNT_EID 72

  ES Set Processor Reset Counter to Zero Command Success Event ID.
- #define CFE_ES_SET_MAX_PR_COUNT_EID 73

  ES Set Maximum Processor Reset Limit Command Success Event ID.
- #define CFE_ES_FILEWRITE_ERR_EID 74

  ES File Write Failed Event ID.
- #define CFE_ES_CDS_DELETE_ERR_EID 76

  ES Delete CDS Command Delete Failed Event ID.
- #define CFE_ES_CDS_NAME_ERR_EID 77

  ES Delete CDS Command Lookup CDS Failed Event ID.
- #define CFE_ES_CDS_DELETED_INFO_EID 78

  ES Delete CDS Command Success Event ID.
- #define CFE_ES_CDS_DELETE_TBL_ERR_EID 79

  ES Delete CDS Command For Critical Table Event ID.
- #define CFE_ES_CDS_OWNER_ACTIVE_EID 80

  ES Delete CDS Command With Active Owner Event ID.
- #define CFE_ES_TLM_POOL_STATS_INFO_EID 81

  ES Telemeter Memory Statistics Command Success Event ID.
- #define CFE_ES_INVALID_POOL_HANDLE_ERR_EID 82

  ES Telemeter Memory Statistics Command Invalid Handle Event ID.
- #define CFE_ES_CDS_REG_DUMP_INF_EID 83

  ES Write Critical Data Store Registry Command Success Event ID.
- #define CFE_ES_CDS_DUMP_ERR_EID 84

  ES Write Critical Data Store Registry Command Record Write Failed Event ID.
- #define CFE_ES_WRITE_CFE_HDR_ERR_EID 85

  ES Write Critical Data Store Registry Command Header Write Failed Event ID.
- #define CFE_ES_CREATING_CDS_DUMP_ERR_EID 86

  ES Write Critical Data Store Registry Command Filename Parse or File Create Failed Event ID.

- #define CFE_ES_TASKINFO_EID 87

  ES Write All Task Data Command Success Event ID.
- #define CFE_ES_TASKINFO_OSCREATE_ERR_EID 88

  ES Write All Task Data Command Filename Parse or File Create Failed Event ID.
- #define CFE_ES_TASKINFO_WRHDR_ERR_EID 89

  ES Write All Task Data Command Write Header Failed Event ID.
- #define CFE_ES_TASKINFO_WR_ERR_EID 90

  ES Write All Task Data Command Write Data Failed Event ID.
- #define CFE_ES_VERSION_INF_EID 91

  cFS Version Information Event ID
- #define CFE_ES_BUILD_INF_EID 92

  cFS Build Information Event ID
- #define CFE_ES_ERLOG_PENDING_ERR_EID 93

  ES Write Exception Reset Log Command Already In Progress Event ID.

12.66.1 Detailed Description

cFE Executive Services Event IDs

12.66.2 Macro Definition Documentation

12.66.2.1 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.

12.66.2.2 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.

12.66.2.3 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.

12.66.2.4 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.

12.66.2.5 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.

12.66.2.6 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.

12.66.2.7 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.

12.66.2.8 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.

12.66.2.9 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.

12.66.2.10 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.

12.66.2.11 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.

12.66.2.12 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.

12.66.2.13 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.

12.66.2.14 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.

12.66.2.15 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.

12.66.2.16 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.

12.66.2.17 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.

12.66.2.18 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.

12.66.2.19 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.

12.66.2.20 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.

12.66.2.21 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.

12.66.2.22 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.

12.66.2.23 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.

12.66.2.24 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.

12.66.2.25 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.

12.66.2.26 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.

12.66.2.27 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.

12.66.2.28 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.

12.66.2.29 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.

12.66.2.30 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.

12.66.2.31 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.

12.66.2.32 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.

12.66.2.33 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.

12.66.2.34 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.

12.66.2.35 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.

12.66.2.36 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.

12.66.2.37 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.

12.66.2.38 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.

12.66.2.39 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.

12.66.2.40 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.

12.66.2.41 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.

12.66.2.42 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.

12.66.2.43 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.

12.66.2.44 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.

12.66.2.45 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.

12.66.2.46 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.

12.66.2.47 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.

12.66.2.48 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.

12.66.2.49 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.

12.66.2.50 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.

12.66.2.51 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 it's associated resources.

Definition at line 510 of file cfe_es_events.h.

12.66.2.52 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.

12.66.2.53 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.

12.66.2.54 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.

12.66.2.55 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.

12.66.2.56 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.

12.66.2.57 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.

12.66.2.58 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.

12.66.2.59 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 it's associated resources.

Definition at line 462 of file cfe_es_events.h.

12.66.2.60 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.

12.66.2.61 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.

12.66.2.62 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.

12.66.2.63 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.

12.66.2.64 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.

12.66.2.65 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.

12.66.2.66 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.

12.66.2.67 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.

12.66.2.68 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.

12.66.2.69 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.

12.66.2.70 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.

12.66.2.71 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.

12.66.2.72 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.

12.66.2.73 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.

12.66.2.74 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.

12.66.2.75 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.

12.66.2.76 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.

12.66.2.77 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.

12.66.2.78 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.

12.66.2.79 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.

12.66.2.80 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.

12.66.2.81 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.

12.66.2.82 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.

12.66.2.83 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.

12.66.2.84 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.

12.66.2.85 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.

12.66.2.86 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.

12.67 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

12.67.1 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:

12.67.2 Macro Definition Documentation

12.67.2.1 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.

12.67.2.2 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.

12.67.2.3 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.

12.67.2.4 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.

12.67.2.5 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.

12.67.2.6 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.

12.67.2.7 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.

12.67.2.8 CFE_ES_QUERY_ALL_TASKS_CC

```
#define CFE_ES_QUERY_ALL_TASKS_CC 24
```

Name Writes a list of All Executive Services Tasks to a File

Description

This command takes the information kept by Executive Services on all of the registered tasks and writes it to the specified file.

Command Mnemonic(s) $sc_$cpu_ES_WriteTaskInfo2File

Command Structure

CFE_ES_QueryAllTasksCmd_t

Command Verification

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

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_TASKINFO_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_TASK↩_LOG_FILE configuration parameter) will be updated with the latest information.

Error Conditions

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

- The file name specified could not be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ONE_CC

Definition at line 1042 of file cfe_es_msg.h.

12.67.2.9 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.

12.67.2.10 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.

12.67.2.11 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.

12.67.2.12 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_ResetPRCnt

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.

12.67.2.13 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.

12.67.2.14 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.

12.67.2.15 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.

12.67.2.16 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_SetMaxPRCnt

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.

12.67.2.17 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_M↩ASK_CC

Definition at line 725 of file cfe_es_msg.h.

12.67.2.18 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.

12.67.2.19 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.

12.67.2.20 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_TRIGG↩ER_MASK_CC

Definition at line 638 of file cfe_es_msg.h.

12.67.2.21 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.

12.67.2.22 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_TRIG↩GER_MASK_CC

Definition at line 688 of file cfe_es_msg.h.

12.67.2.23 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_L↩OG_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.

12.67.2.24 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.

12.67.3 Typedef Documentation

12.67.3.1 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

12.67.3.2 CFE_ES_AppNameCmd_t

```
typedef struct CFE_ES_AppNameCmd CFE_ES_AppNameCmd_t
```

Generic application name command.

12.67.3.3 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

12.67.3.4 CFE_ES_ClearERLogCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLogCmd_t
```

Definition at line 1074 of file cfe_es_msg.h.

12.67.3.5 CFE_ES_ClearSysLogCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSysLogCmd_t
```

Definition at line 1073 of file cfe_es_msg.h.

12.67.3.6 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

12.67.3.7 CFE_ES_DeleteCDSCmd_t

```
typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t
```

Delete Critical Data Store Command.

12.67.3.8 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

12.67.3.9 CFE_ES_DumpCDSRegistryCmd_t

```
typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t
```

Dump CDS Registry Command.

12.67.3.10 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

12.67.3.11 CFE_ES_FileNameCmd_t

```
typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t
```

Generic file name command.

12.67.3.12 CFE_ES_HousekeepingTlm_Payload_t

```
typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t
```

Name Executive Services Housekeeping Packet

12.67.3.13 CFE_ES_HousekeepingTlm_t

```
typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t
```

12.67.3.14 CFE_ES_MemStatsTlm_t

```
typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t
```

12.67.3.15 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)

12.67.3.16 CFE_ES_NoopCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_NoopCmd_t
```

Definition at line 1071 of file cfe_es_msg.h.

12.67.3.17 CFE_ES_OneAppTlm_Payload_t

```
typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t
```

Name Single Application Information Packet

12.67.3.18 CFE_ES_OneAppTlm_t

```
typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t
```

12.67.3.19 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

12.67.3.20 CFE_ES_OverWriteSysLogCmd_t

```
typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_t
```

Overwrite/Discard System Log Configuration Command Payload.

12.67.3.21 CFE_ES_PoolStatsTlm_Payload_t

```
typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t
```

Name Memory Pool Statistics Packet

12.67.3.22 CFE_ES_QueryAllCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllCmd_t
```

Definition at line 1125 of file cfe_es_msg.h.

12.67.3.23 CFE_ES_QueryAllTasksCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasksCmd_t
```

Definition at line 1126 of file cfe_es_msg.h.

12.67.3.24 CFE_ES_QueryOneCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOneCmd_t
```

Definition at line 1211 of file cfe_es_msg.h.

12.67.3.25 CFE_ES_ReloadAppCmd_t

```
typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t
```

Reload Application Command.

12.67.3.26 CFE_ES_ResetCountersCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCountersCmd_t
```

Definition at line 1072 of file cfe_es_msg.h.

12.67.3.27 CFE_ES_ResetPRCountCmd_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCountCmd_t
```

Definition at line 1075 of file cfe_es_msg.h.

12.67.3.28 CFE_ES_RestartAppCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_RestartAppCmd_t
```

Definition at line 1210 of file cfe_es_msg.h.

12.67.3.29 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

12.67.3.30 CFE_ES_RestartCmd_t

```
typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t
```

Restart cFE Command.

12.67.3.31 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

12.67.3.32 CFE_ES_SendMemPoolStatsCmd_t

```
typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t
```

Send Memory Pool Statistics Command.

12.67.3.33 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

12.67.3.34 CFE_ES_SetMaxPRCountCmd_t

```
typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t
```

Set Maximum Processor Reset Count Command.

12.67.3.35 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

12.67.3.36 CFE_ES_SetPerfFilterMaskCmd_t

```
typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t
```

Set Performance Analyzer Filter Mask Command.

12.67.3.37 CFE_ES_SetPerfTriggerMaskCmd_t

```
typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t
```

Set Performance Analyzer Trigger Mask Command.

12.67.3.38 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

12.67.3.39 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

12.67.3.40 CFE_ES_StartAppCmd_t

```
typedef struct CFE_ES_StartApp CFE_ES_StartAppCmd_t
```

Start Application Command.

12.67.3.41 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

12.67.3.42 CFE_ES_StartPerfDataCmd_t

```
typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t
```

Start Performance Analyzer Command.

12.67.3.43 CFE_ES_StopAppCmd_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_StopAppCmd_t
```

Definition at line 1209 of file cfe_es_msg.h.

12.67.3.44 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

12.67.3.45 CFE_ES_StopPerfDataCmd_t

```
typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t
```

Stop Performance Analyzer Command.

12.67.3.46 CFE_ES_WriteERLogCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLogCmd_t
```

Definition at line 1128 of file cfe_es_msg.h.

12.67.3.47 CFE_ES_WriteSysLogCmd_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSysLogCmd_t
```

Definition at line 1127 of file cfe_es_msg.h.

12.68 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_EVTIDNOREGS_EID 6

  EVS Command Event Not Registered For Filtering Event ID.
- #define CFE_EVS_ERR_APPNOREGS_EID 7

  EVS Command Application Not Registered With EVS Event ID.
- #define CFE_EVS_ERR_ILLAPPIDRANGE_EID 8

  EVS Command Get Application Data Failure Event ID.
- #define CFE_EVS_ERR_NOAPPIDFOUND_EID 9

  EVS Command Get Application ID Failure Event ID.
- #define CFE_EVS_ERR_ILLEGALFMTMOD_EID 10

  EVS Set Event Format Command Invalid Format Event ID.
- #define CFE_EVS_ERR_MAXREGSFILTER_EID 11

  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.

12.68.1 Detailed Description

cFE Event Services Event IDs

12.68.2 Macro Definition Documentation

12.68.2.1 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.

12.68.2.2 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.

12.68.2.3 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.

12.68.2.4 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.

12.68.2.5 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.

12.68.2.6 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.

12.68.2.7 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.

12.68.2.8 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.

12.68.2.9 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.

12.68.2.10 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.

12.68.2.11 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.

12.68.2.12 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.

12.68.2.13 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.

12.68.2.14 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.

12.68.2.15 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.

12.68.2.16 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.

12.68.2.17 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.

12.68.2.18 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.

12.68.2.19 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.

12.68.2.20 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.

12.68.2.21 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.

12.68.2.22 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.

12.68.2.23 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.

12.68.2.24 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.

12.68.2.25 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.

12.68.2.26 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.

12.68.2.27 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.

12.68.2.28 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.

12.68.2.29 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.

12.68.2.30 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.

12.68.2.31 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.

12.68.2.32 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.

12.68.2.33 CFE_EVS_RSTEVTCNT_EID

```
#define CFE_EVS_RSTEVTCNT_EID 27
```

EVS Reset App Event Counter Command Success Event ID.

Type: DEBUG

Cause:

EVS Reset App Event Counter Command success.

Definition at line 323 of file cfe_evs_events.h.

12.68.2.34 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.

12.68.2.35 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.

12.68.2.36 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.

12.68.2.37 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.

12.68.2.38 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.

12.68.2.39 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.

12.69 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_WriteLogDataFileCmd

  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_WriteLogDataFileCmd CFE_EVS_WriteLogDataFileCmd_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_↩ Payload_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

12.69.1 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

12.69.2 Macro Definition Documentation

12.69.2.1 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.

12.69.2.2 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.

12.69.2.3 CFE_EVS_CRITICAL_BIT

```
#define CFE_EVS_CRITICAL_BIT 0x0008
```

Definition at line 892 of file cfe_evs_msg.h.

12.69.2.4 CFE_EVS_DEBUG_BIT

```
#define CFE_EVS_DEBUG_BIT 0x0001
```

Definition at line 889 of file cfe_evs_msg.h.

12.69.2.5 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.

12.69.2.6 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_AP↩P_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.

12.69.2.7 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_AP↩P_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.

12.69.2.8 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_DISABL↩E_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.

12.69.2.9 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.

12.69.2.10 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_AP↩P_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.

12.69.2.11 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_AP↩P_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.

12.69.2.12 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_DISABL↩E_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.

12.69.2.13 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.

12.69.2.14 CFE_EVS_ERROR_BIT

```
#define CFE_EVS_ERROR_BIT 0x0004
```

Definition at line 891 of file cfe_evs_msg.h.

12.69.2.15 CFE_EVS_INFORMATION_BIT

```
#define CFE_EVS_INFORMATION_BIT 0x0002
```

Definition at line 890 of file cfe_evs_msg.h.

12.69.2.16 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.

12.69.2.17 CFE_EVS_PORT1_BIT

```
#define CFE_EVS_PORT1_BIT 0x0001
```

Definition at line 895 of file cfe_evs_msg.h.

12.69.2.18 CFE_EVS_PORT2_BIT

```
#define CFE_EVS_PORT2_BIT 0x0002
```

Definition at line 896 of file cfe_evs_msg.h.

12.69.2.19 CFE_EVS_PORT3_BIT

```
#define CFE_EVS_PORT3_BIT 0x0004
```

Definition at line 897 of file cfe_evs_msg.h.

12.69.2.20 CFE_EVS_PORT4_BIT

```
#define CFE_EVS_PORT4_BIT 0x0008
```

Definition at line 898 of file cfe_evs_msg.h.

12.69.2.21 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.

12.69.2.22 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.

12.69.2.23 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.

12.69.2.24 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.

12.69.2.25 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.

12.69.2.26 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.

12.69.2.27 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_DI↩SCARD

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.

12.69.2.28 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.

12.69.2.29 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_WriteLogDataFileCmd_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.

12.69.3 Typedef Documentation

12.69.3.1 CFE_EVS_AddEventFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_AddEventFilterCmd_t
```

Definition at line 1151 of file cfe_evs_msg.h.

12.69.3.2 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

12.69.3.3 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

12.69.3.4 CFE_EVS_AppNameBitMaskCmd_t

```
typedef struct CFE_EVS_AppNameBitMaskCmd CFE_EVS_AppNameBitMaskCmd_t
```

Generic App Name and Bitmask Command.

12.69.3.5 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

12.69.3.6 CFE_EVS_AppNameCmd_t

```
typedef struct CFE_EVS_AppNameCmd CFE_EVS_AppNameCmd_t
```

Generic App Name Command.

12.69.3.7 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

12.69.3.8 CFE_EVS_AppNameEventIDCmd_t

```
typedef struct CFE_EVS_AppNameEventIDCmd CFE_EVS_AppNameEventIDCmd_t
```

Generic App Name and Event ID Command.

12.69.3.9 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

12.69.3.10 CFE_EVS_AppNameEventIDMaskCmd_t

```
typedef struct CFE_EVS_AppNameEventIDMaskCmd CFE_EVS_AppNameEventIDMaskCmd_t
```

Generic App Name, Event ID, Mask Command.

12.69.3.11 CFE_EVS_AppTlmData_t

```
typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t
```

12.69.3.12 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

12.69.3.13 CFE_EVS_BitMaskCmd_t

```
typedef struct CFE_EVS_BitMaskCmd CFE_EVS_BitMaskCmd_t
```

Generic Bitmask Command.

12.69.3.14 CFE_EVS_ClearLogCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLogCmd_t
```

Definition at line 917 of file cfe_evs_msg.h.

12.69.3.15 CFE_EVS_DeleteEventFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilterCmd_t
```

Definition at line 1091 of file cfe_evs_msg.h.

12.69.3.16 CFE_EVS_DisableAppEventsCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEventsCmd_t
```

Definition at line 1060 of file cfe_evs_msg.h.

12.69.3.17 CFE_EVS_DisableAppEventTypeCmd_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventTypeCmd_t
```

Definition at line 1121 of file cfe_evs_msg.h.

12.69.3.18 CFE_EVS_DisableEventTypeCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventTypeCmd_t
```

Definition at line 1031 of file cfe_evs_msg.h.

12.69.3.19 CFE_EVS_DisablePortsCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePortsCmd_t
```

Definition at line 1029 of file cfe_evs_msg.h.

12.69.3.20 CFE_EVS_EnableAppEventsCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEventsCmd_t
```

Definition at line 1059 of file cfe_evs_msg.h.

12.69.3.21 CFE_EVS_EnableAppEventTypeCmd_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventTypeCmd_t
```

Definition at line 1120 of file cfe_evs_msg.h.

12.69.3.22 CFE_EVS_EnableEventTypeCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventTypeCmd_t
```

Definition at line 1030 of file cfe_evs_msg.h.

12.69.3.23 CFE_EVS_EnablePortsCmd_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePortsCmd_t
```

Definition at line 1028 of file cfe_evs_msg.h.

12.69.3.24 CFE_EVS_HousekeepingTlm_Payload_t

```
typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t
```

Name Event Services Housekeeping Telemetry Packet

12.69.3.25 CFE_EVS_HousekeepingTlm_t

```
typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t
```

12.69.3.26 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

12.69.3.27 CFE_EVS_LongEventTlm_Payload_t

```
typedef struct CFE_EVS_LongEventTlm_Payload CFE_EVS_LongEventTlm_Payload_t
```

Name Event Message Telemetry Packet (Long format)

12.69.3.28 CFE_EVS_LongEventTlm_t

```
typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t
```

12.69.3.29 CFE_EVS_NoArgsCmd_t

```
typedef struct CFE_EVS_NoArgsCmd CFE_EVS_NoArgsCmd_t
```

Command with no additional arguments.

12.69.3.30 CFE_EVS_NoopCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_NoopCmd_t
```

Definition at line 915 of file cfe_evs_msg.h.

12.69.3.31 CFE_EVS_PacketID_t

```
typedef struct CFE_EVS_PacketID CFE_EVS_PacketID_t
```

Telemetry packet structures

12.69.3.32 CFE_EVS_ResetAllFiltersCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFiltersCmd_t
```

Definition at line 1062 of file cfe_evs_msg.h.

12.69.3.33 CFE_EVS_ResetAppCounterCmd_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounterCmd_t
```

Definition at line 1061 of file cfe_evs_msg.h.

12.69.3.34 CFE_EVS_ResetCountersCmd_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCountersCmd_t
```

Definition at line 916 of file cfe_evs_msg.h.

12.69.3.35 CFE_EVS_ResetFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilterCmd_t
```

Definition at line 1090 of file cfe_evs_msg.h.

12.69.3.36 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

12.69.3.37 CFE_EVS_SetEventFormatModeCmd_t

```
typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t
```

Set Event Format Mode Command.

12.69.3.38 CFE_EVS_SetFilterCmd_t

```
typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilterCmd_t
```

Definition at line 1152 of file cfe_evs_msg.h.

12.69.3.39 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

12.69.3.40 CFE_EVS_SetLogModeCmd_t

```
typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t
```

Set Log Mode Command.

12.69.3.41 CFE_EVS_ShortEventTlm_Payload_t

```
typedef struct CFE_EVS_ShortEventTlm_Payload CFE_EVS_ShortEventTlm_Payload_t
```

Name Event Message Telemetry Packet (Short format)

12.69.3.42 CFE_EVS_ShortEventTlm_t

```
typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t
```

12.69.3.43 CFE_EVS_WriteAppDataFileCmd_t

```
typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd_t
```

Write Event Services Application Information to File Command.

12.69.3.44 CFE_EVS_WriteLogDataFileCmd_t

```
typedef struct CFE_EVS_WriteLogDataFileCmd CFE_EVS_WriteLogDataFileCmd_t
```

Write Event Log to File Command.

12.70 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.

12.70.1 Detailed Description

Define CCSDS packet header types

- Avoid direct access for portability, use APIs
- Used to construct message structures

12.70.2 Typedef Documentation

12.70.2.1 CCSDS_ExtendedHeader_t

```
typedef struct CCSDS_ExtendedHeader CCSDS_ExtendedHeader_t
```

CCSDS packet extended header.

12.70.2.2 CCSDS_PrimaryHeader_t

```
typedef struct CCSDS_PrimaryHeader CCSDS_PrimaryHeader_t
```

CCSDS packet primary header.

12.71 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_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) }

12.71.1 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.

12.72 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.

12.72.1 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.

12.72.2 Macro Definition Documentation

12.72.2.1 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.

12.72.2.2 CFE_RESOURCEID_MAX

```
#define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK
```

Definition at line 65 of file cfe_resourceid_basevalue.h.

12.72.2.3 CFE_RESOURCEID_SHIFT

```
#define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT
```

Definition at line 64 of file cfe_resourceid_basevalue.h.

12.73 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 Not Owner Event ID.
- #define CFE_SB_SETPIPEOPTS_EID 57

  SB Set Pipe Opts API Success Event ID.
- #define CFE_SB_GETPIPEOPTS_ID_ERR_EID 58

  SB Get Pipe Opts API Invalid Pipe Event ID.
- #define CFE_SB_GETPIPEOPTS_PTR_ERR_EID 59

  SB Get Pipe Opts API Invalid Pointer Event ID.
- #define CFE_SB_GETPIPEOPTS_EID 60

  SB Get Pipe Opts API Success Event ID.
- #define CFE_SB_GETPIPENAME_EID 62

  SB Get Pipe Name API Success Event ID.
- #define CFE_SB_GETPIPENAME_NULL_PTR_EID 63

  SB Get Pipe Name API Invalid Pointer Event ID.
- #define CFE_SB_GETPIPENAME_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.

12.73.1 Detailed Description

cFE Software Bus Services Event IDs

12.73.2 Macro Definition Documentation

12.73.2.1 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.

12.73.2.2 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.

12.73.2.3 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.

12.73.2.4 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.

12.73.2.5 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.

12.73.2.6 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.

12.73.2.7 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.

12.73.2.8 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.

12.73.2.9 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.

12.73.2.10 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.

12.73.2.11 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.

12.73.2.12 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.

12.73.2.13 CFE_SB_DSBL_RTE1_EID

```
#define CFE_SB_DSBL_RTE1_EID 36
```

SB Disable Route Command Invalid MsgId/PipeId Pair Event ID.

Type: ERROR

Cause:

SB Disable Route Command failure due to the Message ID not being subscribed to the pipe.

Definition at line 404 of file cfe_sb_events.h.

12.73.2.14 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.

12.73.2.15 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.

12.73.2.16 CFE_SB_DUP_SUBSCRIP_EID

```
#define CFE_SB_DUP_SUBSCRIP_EID 7
```

SB Subscribe API Duplicate MsgId Subscription Event ID.

Type: INFORMATION

Cause:

An SB Subscribe API was called with a Message ID that was already subscribed on the pipe on the pipe.

Definition at line 109 of file cfe_sb_events.h.

12.73.2.17 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.

12.73.2.18 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.

12.73.2.19 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.

12.73.2.20 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.

12.73.2.21 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.

12.73.2.22 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.

12.73.2.23 CFE_SB_GETPIPEIDBYNAME_EID

```
#define CFE_SB_GETPIPEIDBYNAME_EID 65
```

SB Get Pipe ID By Name API Success Event ID.

Type: DEBUG

Cause:

CFE_SB_GetPipeIdByName success.

Definition at line 705 of file cfe_sb_events.h.

12.73.2.24 CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID

```
#define CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID 67
```

SB Get Pipe ID By Name API Name Not Found Or ID Not Matched Event ID.

Type: ERROR

Cause:

CFE_SB_GetPipeIdByName failure due to name not found or ID mismatch. OVERLOADED

Definition at line 727 of file cfe_sb_events.h.

12.73.2.25 CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID

```
#define CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID 66
```

SB Get Pipe ID By Name API Invalid Pointer Event ID.

Type: ERROR

Cause:

CFE_SB_GetPipeIdByName failure due to invalid pointer.

Definition at line 716 of file cfe_sb_events.h.

12.73.2.26 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.

12.73.2.27 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.

12.73.2.28 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.

12.73.2.29 CFE_SB_GETPIPEOPTS_EID

```
#define CFE_SB_GETPIPEOPTS_EID 60
```

SB Get Pipe Opts API Success Event ID.

Type: DEBUG

Cause:

CFE_SB_GetPipeOpts success.

Definition at line 661 of file cfe_sb_events.h.

12.73.2.30 CFE_SB_GETPIPEOPTS_ID_ERR_EID

```
#define CFE_SB_GETPIPEOPTS_ID_ERR_EID 58
```

SB Get Pipe Opts 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.

12.73.2.31 CFE_SB_GETPIPEOPTS_PTR_ERR_EID

```
#define CFE_SB_GETPIPEOPTS_PTR_ERR_EID 59
```

SB Get Pipe Opts 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.

12.73.2.32 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.

12.73.2.33 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.

12.73.2.34 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.

12.73.2.35 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.

12.73.2.36 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.

12.73.2.37 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.

12.73.2.38 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.

12.73.2.39 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.

12.73.2.40 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.

12.73.2.41 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.

12.73.2.42 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.

12.73.2.43 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.

12.73.2.44 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.

12.73.2.45 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.

12.73.2.46 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.

12.73.2.47 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.

12.73.2.48 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.

12.73.2.49 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.

12.73.2.50 CFE_SB_SETPIPEOPTS_EID

```
#define CFE_SB_SETPIPEOPTS_EID 57
```

SB Set Pipe Opts API Success Event ID.

Type: DEBUG

Cause:

CFE_SB_SetPipeOpts success.

Definition at line 628 of file cfe_sb_events.h.

12.73.2.51 CFE_SB_SETPIPEOPTS_ID_ERR_EID

```
#define CFE_SB_SETPIPEOPTS_ID_ERR_EID 55
```

SB Set Pipe Opts API Invalid Pipe Event ID.

Type: ERROR

Cause:

CFE_SB_SetPipeOpts API failure due to an invalid pipe ID

Definition at line 606 of file cfe_sb_events.h.

12.73.2.52 CFE_SB_SETPIPEOPTS_OWNER_ERR_EID

```
#define CFE_SB_SETPIPEOPTS_OWNER_ERR_EID 56
```

SB Set Pipe Opts API Not Owner Event ID.

Type: ERROR

Cause:

CFE_SB_SetPipeOpts API failure due to not being the pipe owner.

Definition at line 617 of file cfe_sb_events.h.

12.73.2.53 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.

12.73.2.54 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.

12.73.2.55 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.

12.73.2.56 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.

12.73.2.57 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.

12.73.2.58 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.

12.73.2.59 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.

12.73.2.60 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.

12.73.2.61 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.

12.73.2.62 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.

12.73.2.63 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.

12.73.2.64 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.

12.73.2.65 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.

12.74 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

12.74.1 Detailed Description

Purpose: This header file contains structure definitions for all SB command and telemetry packets

Author: R.McGraw/SSI

12.74.2 Macro Definition Documentation

12.74.2.1 CFE_SB_DISABLE_ROUTE_CC

```
#define CFE_SB_DISABLE_ROUTE_CC 5
```

Name Disable Software Bus Route

Description

This command will disable a particular destination. The destination is specified in terms of MsgID and PipeID. The MsgId and PipeID are parameters in the command. All destinations are enabled by default.

Command Mnemonic(s) $sc_$cpu_SB_DisRoute

Command Structure

CFE_SB_DisableRouteCmd_t

Command Verification

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

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- View routing information CFE_SB_WRITE_ROUTING_INFO_CC to verify enable/disable state change
- The CFE_SB_DSBL_RTE2_EID debug event message will be generated
- Destination will stop receiving messages

Error Conditions

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

- the MsgId or PipeId parameters do not pass validation
- the destination does not exist.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_DSBL_RTE1_EID or CFE_SB_DSBL_RTE3_EID

Criticality

This command is not intended to be used in nominal conditions. It is possible to get into a state where a destination cannot be re-enabled without resetting the processor. For instance, sending this command with CFE_SB_CMD_MID and the SB_Cmd_Pipe would inhibit any ground commanding to the software bus until the processor was reset. There are similar problems that may occur when using this command.

Definition at line 281 of file cfe_sb_msg.h.

12.74.2.2 CFE_SB_DISABLE_SUB_REPORTING_CC

```
#define CFE_SB_DISABLE_SUB_REPORTING_CC 10
```

Name Disable Subscription Reporting Command

Description

This command will disable subscription reporting and is intended to be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is needed by SBN if offboard routing is required.

Command Mnemonic(s) $sc_$cpu_SB_DisSubRptg

Command Structure

CFE_SB_DisableSubReportingCmd_t

Command Verification

Successful execution of this command will result in the suppression of packets (with the CFE_SB_ONESUB_TLM_MID MsgId) for each subscription received by SB through the subscription APIs.

Error Conditions

None

Criticality

None

See also

CFE_SB_SingleSubscriptionTlm_t, CFE_SB_ENABLE_SUB_REPORTING_CC, CFE_SB_SEND_PREV_SUBS_CC

Definition at line 438 of file cfe_sb_msg.h.

12.74.2.3 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.

12.74.2.4 CFE_SB_ENABLE_SUB_REPORTING_CC

```
#define CFE_SB_ENABLE_SUB_REPORTING_CC 9
```

Name Enable Subscription Reporting Command

Description

This command will enable subscription reporting and is intended to be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is needed by SBN if offboard routing is required.

Command Mnemonic(s) $sc_$cpu_SB_EnaSubRptg

Command Structure

CFE_SB_EnableSubReportingCmd_t

Command Verification

Successful execution of this command will result in the sending of a packet (with the CFE_SB_ONESUB_TLM_MID MsgId) for each subscription received by SB through the subscription APIs.

Error Conditions

None

Criticality

None

See also

CFE_SB_SingleSubscriptionTlm_t, CFE_SB_DISABLE_SUB_REPORTING_CC, CFE_SB_SEND_PREV_SUBS_CC

Definition at line 405 of file cfe_sb_msg.h.

12.74.2.5 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.

12.74.2.6 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.

12.74.2.7 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 S↩BN(Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When this command is received the software bus will generate and send a series of packets containing information about all subscription previously received.

Command Mnemonic(s) $sc_$cpu_SB_SendPrevSubs

Command Structure

CFE_SB_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_REP↩ORTING_CC

Definition at line 470 of file cfe_sb_msg.h.

12.74.2.8 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.

12.74.2.9 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_DE↩FAULT_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.

12.74.2.10 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.

12.74.2.11 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.

12.74.3 Typedef Documentation

12.74.3.1 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.

12.74.3.2 CFE_SB_AllSubscriptionsTlm_t

```
typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t
```

12.74.3.3 CFE_SB_DisableRouteCmd_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_DisableRouteCmd_t
```

Definition at line 545 of file cfe_sb_msg.h.

12.74.3.4 CFE_SB_DisableSubReportingCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_DisableSubReportingCmd_t
```

Definition at line 486 of file cfe_sb_msg.h.

12.74.3.5 CFE_SB_EnableRouteCmd_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_EnableRouteCmd_t
```

Definition at line 544 of file cfe_sb_msg.h.

12.74.3.6 CFE_SB_EnableSubReportingCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_EnableSubReportingCmd_t
```

Definition at line 485 of file cfe_sb_msg.h.

12.74.3.7 CFE_SB_HousekeepingTlm_Payload_t

```
typedef struct CFE_SB_HousekeepingTlm_Payload CFE_SB_HousekeepingTlm_Payload_t
```

Name Software Bus task housekeeping Packet

12.74.3.8 CFE_SB_HousekeepingTlm_t

```
typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t
```

12.74.3.9 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

12.74.3.10 CFE_SB_NoopCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_NoopCmd_t
```

Definition at line 483 of file cfe_sb_msg.h.

12.74.3.11 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

12.74.3.12 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.

12.74.3.13 CFE_SB_ResetCountersCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd_t
```

Definition at line 484 of file cfe_sb_msg.h.

12.74.3.14 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 PipeId combination.

12.74.3.15 CFE_SB_RouteCmd_t

```
typedef struct CFE_SB_RouteCmd CFE_SB_RouteCmd_t
```

Enable/Disable Route Command.

12.74.3.16 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

12.74.3.17 CFE_SB_SendPrevSubsCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_SendPrevSubsCmd_t
```

Definition at line 488 of file cfe_sb_msg.h.

12.74.3.18 CFE_SB_SendSbStatsCmd_t

```
typedef CFE_MSG_CommandHeader_t CFE_SB_SendSbStatsCmd_t
```

Definition at line 487 of file cfe_sb_msg.h.

12.74.3.19 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

12.74.3.20 CFE_SB_SingleSubscriptionTlm_t

```
typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscriptionTlm_t
```

12.74.3.21 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

12.74.3.22 CFE_SB_StatsTlm_t

```
typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t
```

12.74.3.23 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

12.74.3.24 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

12.74.3.25 CFE_SB_WriteFileInfoCmd_t

```
typedef struct CFE_SB_WriteFileInfoCmd CFE_SB_WriteFileInfoCmd_t
```

Write File Info Command.

12.74.3.26 CFE_SB_WriteMapInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteMapInfoCmd_t
```

Definition at line 514 of file cfe_sb_msg.h.

12.74.3.27 CFE_SB_WritePipeInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WritePipeInfoCmd_t
```

Definition at line 513 of file cfe_sb_msg.h.

12.74.3.28 CFE_SB_WriteRoutingInfoCmd_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteRoutingInfoCmd_t
```

Definition at line 512 of file cfe_sb_msg.h.

12.75 cfe/modules/tbl/fsw/inc/cfe_tbl_events.h File Reference

Macros

TBL event IDs

- #define CFE_TBL_INIT_INF_EID 1

  TB Initialization Event ID.
- #define CFE_TBL_NOOP_INF_EID 10

  TBL No-op Command Success Event ID.
- #define CFE_TBL_RESET_INF_EID 11

  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

TBL Load Table File Open Failure Event ID.

- #define CFE_TBL_FILE_STD_HDR_ERR_EID 54

  TBL Load Table File Read Standard Header Failure Event ID.
- #define CFE_TBL_FILE_TBL_HDR_ERR_EID 55

  TBL Load Table File Read Table Header Failure Event ID.
- #define CFE_TBL_FAIL_HK_SEND_ERR_EID 56

  TBL Send Housekeeping Command Transmit Failure Event ID.
- #define CFE_TBL_NO_SUCH_TABLE_ERR_EID 57

  TBL Table Name Not Found Event ID.
- #define CFE_TBL_FILE_TYPE_ERR_EID 58

  TBL Load Table Invalid File Content ID Event ID.
- #define CFE_TBL_FILE_SUBTYPE_ERR_EID 59

  TBL Load Table Invalid File Subtype Event ID.
- #define CFE_TBL_NO_WORK_BUFFERS_ERR_EID 60

  TBL Load Or Dump Table No Working Buffers Available Event ID.
- #define CFE_TBL_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.

12.75.1 Detailed Description

cFE Table Services Event IDs

12.75.2 Macro Definition Documentation

12.75.2.1 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.

12.75.2.2 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.

12.75.2.3 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.

12.75.2.4 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.

12.75.2.5 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.

12.75.2.6 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.

12.75.2.7 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.

12.75.2.8 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.

12.75.2.9 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.

12.75.2.10 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.

12.75.2.11 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.

12.75.2.12 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.

12.75.2.13 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.

12.75.2.14 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.

12.75.2.15 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.

12.75.2.16 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.

12.75.2.17 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.

12.75.2.18 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.

12.75.2.19 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.

12.75.2.20 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.

12.75.2.21 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.

12.75.2.22 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.

12.75.2.23 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.

12.75.2.24 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.

12.75.2.25 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.

12.75.2.26 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.

12.75.2.27 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.

12.75.2.28 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.

12.75.2.29 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.

12.75.2.30 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.

12.75.2.31 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.

12.75.2.32 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.

12.75.2.33 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.

12.75.2.34 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.

12.75.2.35 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.

12.75.2.36 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.

12.75.2.37 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.

12.75.2.38 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.

12.75.2.39 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.

12.75.2.40 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.

12.75.2.41 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.

12.75.2.42 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.

12.75.2.43 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.

12.75.2.44 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.

12.75.2.45 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.

12.75.2.46 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.

12.75.2.47 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.

12.75.2.48 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.

12.75.2.49 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.

12.75.2.50 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.

12.75.2.51 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.

12.75.2.52 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.

12.75.2.53 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.

12.75.2.54 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 housekeeping packet.

Definition at line 146 of file cfe_tbl_events.h.

12.75.2.55 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.

12.75.2.56 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.

12.75.2.57 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.

12.75.2.58 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.

12.75.2.59 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.

12.75.2.60 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.

12.75.2.61 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.

12.75.2.62 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.

12.75.2.63 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.

12.75.2.64 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.

12.75.2.65 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.

12.75.2.66 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.

12.75.2.67 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.

12.75.2.68 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.

12.75.2.69 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.

12.75.2.70 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.

12.76 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_DelCDSCmd_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_DelCDSCmd_Payload CFE_TBL_DelCDSCmd_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

12.76.1 Detailed Description

Purpose: cFE Table Services (TBL) SB message definitions header file

Author: D.Kobe/Hammers

Notes:

12.76.2 Macro Definition Documentation

12.76.2.1 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.

12.76.2.2 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.

12.76.2.3 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.

12.76.2.4 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.

12.76.2.5 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.

12.76.2.6 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.

12.76.2.7 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.

12.76.2.8 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.

12.76.2.9 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.

12.76.2.10 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.

12.76.3 Typedef Documentation

12.76.3.1 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

12.76.3.2 CFE_TBL_AbortLoadCmd_t

```
typedef struct CFE_TBL_AbortLoadCmd CFE_TBL_AbortLoadCmd_t
```

Abort Load Command.

12.76.3.3 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

12.76.3.4 CFE_TBL_ActivateCmd_t

```
typedef struct CFE_TBL_ActivateCmd CFE_TBL_ActivateCmd_t
```

Activate Table Command.

12.76.3.5 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

12.76.3.6 CFE_TBL_DeleteCDSCmd_t

```
typedef struct CFE_TBL_DeleteCDSCmd CFE_TBL_DeleteCDSCmd_t
```

Delete Critical Table CDS Command.

12.76.3.7 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

12.76.3.8 CFE_TBL_DumpCmd_t

```
typedef struct CFE_TBL_DumpCmd CFE_TBL_DumpCmd_t
```

/brief Dump Table Command

12.76.3.9 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

12.76.3.10 CFE_TBL_DumpRegistryCmd_t

```
typedef struct CFE_TBL_DumpRegistryCmd CFE_TBL_DumpRegistryCmd_t
```

Dump Registry Command.

12.76.3.11 CFE_TBL_HousekeepingTlm_Payload_t

```
typedef struct CFE_TBL_HousekeepingTlm_Payload CFE_TBL_HousekeepingTlm_Payload_t
```

Name Table Services Housekeeping Packet

12.76.3.12 CFE_TBL_HousekeepingTlm_t

```
typedef struct CFE_TBL_HousekeepingTlm CFE_TBL_HousekeepingTlm_t
```

12.76.3.13 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

12.76.3.14 CFE_TBL_LoadCmd_t

```
typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t
```

Load Table Command.

12.76.3.15 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)

12.76.3.16 CFE_TBL_NoopCmd_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_NoopCmd_t
```

Definition at line 500 of file cfe_tbl_msg.h.

12.76.3.17 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).

12.76.3.18 CFE_TBL_NotifyCmd_t

```
typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t
```

/brief Table Management Notification Command

12.76.3.19 CFE_TBL_ResetCountersCmd_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_ResetCountersCmd_t
```

Definition at line 501 of file cfe_tbl_msg.h.

12.76.3.20 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

12.76.3.21 CFE_TBL_SendRegistryCmd_t

```
typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t
```

Send Table Registry Command.

12.76.3.22 CFE_TBL_TableRegistryTlm_t

```
typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t
```

12.76.3.23 CFE_TBL_TblRegPacket_Payload_t

```
typedef struct CFE_TBL_TblRegPacket_Payload CFE_TBL_TblRegPacket_Payload_t
```

Name Table Registry Info Packet

12.76.3.24 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

12.76.3.25 CFE_TBL_ValidateCmd_t

```
typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t
```

Validate Table Command.

12.77 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.

12.77.1 Detailed Description

cFE Time Services Event IDs

12.77.2 Macro Definition Documentation

12.77.2.1 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.

12.77.2.2 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.

12.77.2.3 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.

12.77.2.4 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.

12.77.2.5 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.

12.77.2.6 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.

12.77.2.7 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.

12.77.2.8 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.

12.77.2.9 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.

12.77.2.10 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.

12.77.2.11 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.

12.77.2.12 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.

12.77.2.13 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.

12.77.2.14 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.

12.77.2.15 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.

12.77.2.16 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.

12.77.2.17 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.

12.77.2.18 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.

12.77.2.19 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.

12.77.2.20 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.

12.77.2.21 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.

12.77.2.22 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.

12.77.2.23 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.

12.77.2.24 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.

12.77.2.25 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.

12.77.2.26 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.

12.77.2.27 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.

12.77.2.28 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.

12.77.2.29 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.

12.77.2.30 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.

12.77.2.31 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.

12.77.2.32 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.

12.77.2.33 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.

12.77.2.34 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.

12.77.2.35 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.

12.77.2.36 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.

12.78 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

12.78.1 Detailed Description

Purpose: cFE Time Services (TIME) SB message definitions header file

Author: S.Walling/Microtel

Notes:

12.78.2 Macro Definition Documentation

12.78.2.1 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.

12.78.2.2 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.

12.78.2.3 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.

12.78.2.4 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.

12.78.2.5 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.

12.78.2.6 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.

12.78.2.7 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.

12.78.2.8 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.

12.78.2.9 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_Tone↩SignalSelect_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.

12.78.2.10 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_Source↩ Select_EXTERNAL was specified)
- Time source selection not allowed on this platform

  Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TIME_CMDEC`** - Command Error counter will increment
- Error specific event message (either CFE_TIME_SOURCE_CFG_EID or CFE_TIME_SOURCE_ERR_EID)

Criticality

Although clock source selection is important, this command is not critical.

See also

CFE_TIME_SET_STATE_CC, CFE_TIME_SET_SIGNAL_CC

Definition at line 206 of file cfe_time_msg.h.

12.78.2.11 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↩ 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.

12.78.2.12 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.

12.78.2.13 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.

12.78.2.14 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.

12.78.2.15 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_ADJUST↩MENT_CC

Definition at line 566 of file cfe_time_msg.h.

12.78.2.16 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_DLatentSs - Housekeeping Telemetry point indicating command specified values
- **`$sc_$cpu_TIME_DLatentDir`** - Diagnostic Telemetry point indicating commanded latency direction
- The CFE_TIME_DELAY_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Client

  Evidence of Failure may be found in the following telemetry:
- **`$sc_$cpu_TIME_CMDEC`** - command error counter will increment
- Error specific event messages will be issued (CFE_TIME_DELAY_CFG_EID or CFE_TIME_DELAY_ERR↩_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_DELAY_CC

Definition at line 339 of file cfe_time_msg.h.

12.78.3 Typedef Documentation

12.78.3.1 CFE_TIME_1HzCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_1HzCmd_t
```

Definition at line 743 of file cfe_time_msg.h.

12.78.3.2 CFE_TIME_Add1HZAdjustmentCmd_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustmentCmd_t
```

Definition at line 878 of file cfe_time_msg.h.

12.78.3.3 CFE_TIME_AddAdjustCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjustCmd_t
```

Definition at line 850 of file cfe_time_msg.h.

12.78.3.4 CFE_TIME_AddDelayCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelayCmd_t
```

Definition at line 846 of file cfe_time_msg.h.

12.78.3.5 CFE_TIME_DiagnosticTlm_Payload_t

```
typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t
```

Name Time Services Diagnostics Packet

12.78.3.6 CFE_TIME_DiagnosticTlm_t

```
typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t
```

12.78.3.7 CFE_TIME_FakeToneCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_FakeToneCmd_t
```

Definition at line 745 of file cfe_time_msg.h.

12.78.3.8 CFE_TIME_HousekeepingTlm_Payload_t

```
typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t
```

Name Time Services Housekeeping Packet

12.78.3.9 CFE_TIME_HousekeepingTlm_t

```
typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t
```

12.78.3.10 CFE_TIME_LeapsCmd_Payload_t

```
typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t
```

Set leap seconds command payload.

12.78.3.11 CFE_TIME_NoArgsCmd_t

```
typedef struct CFE_TIME_NoArgsCmd CFE_TIME_NoArgsCmd_t
```

Generic no argument command.

12.78.3.12 CFE_TIME_NoopCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_NoopCmd_t
```

Definition at line 740 of file cfe_time_msg.h.

12.78.3.13 CFE_TIME_OneHzAdjustmentCmd_Payload_t

```
typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t
```

Generic seconds, subseconds command payload.

12.78.3.14 CFE_TIME_OneHzAdjustmentCmd_t

```
typedef struct CFE_TIME_OneHzAdjustmentCmd CFE_TIME_OneHzAdjustmentCmd_t
```

Generic seconds, subseconds adjustment command.

12.78.3.15 CFE_TIME_ResetCountersCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCountersCmd_t
```

Definition at line 741 of file cfe_time_msg.h.

12.78.3.16 CFE_TIME_SendDiagnosticCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticCmd_t
```

Definition at line 742 of file cfe_time_msg.h.

12.78.3.17 CFE_TIME_SetLeapSecondsCmd_t

```
typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t
```

Set leap seconds command.

12.78.3.18 CFE_TIME_SetMETCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMETCmd_t
```

Definition at line 848 of file cfe_time_msg.h.

12.78.3.19 CFE_TIME_SetSignalCmd_t

```
typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t
```

Set tone signal source command.

12.78.3.20 CFE_TIME_SetSourceCmd_t

```
typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t
```

Set time data source command.

12.78.3.21 CFE_TIME_SetStateCmd_t

```
typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t
```

Set clock state command.

12.78.3.22 CFE_TIME_SetSTCFCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCFCmd_t
```

Definition at line 849 of file cfe_time_msg.h.

12.78.3.23 CFE_TIME_SetTimeCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTimeCmd_t
```

Definition at line 852 of file cfe_time_msg.h.

12.78.3.24 CFE_TIME_SignalCmd_Payload_t

```
typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t
```

Set tone signal source command payload.

12.78.3.25 CFE_TIME_SourceCmd_Payload_t

```
typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t
```

Set time data source command payload.

12.78.3.26 CFE_TIME_StateCmd_Payload_t

```
typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t
```

Set clock state command payload.

12.78.3.27 CFE_TIME_Sub1HZAdjustmentCmd_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustmentCmd_t
```

Definition at line 879 of file cfe_time_msg.h.

12.78.3.28 CFE_TIME_SubAdjustCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjustCmd_t
```

Definition at line 851 of file cfe_time_msg.h.

12.78.3.29 CFE_TIME_SubDelayCmd_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelayCmd_t
```

Definition at line 847 of file cfe_time_msg.h.

12.78.3.30 CFE_TIME_TimeCmd_Payload_t

```
typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t
```

Generic seconds, microseconds command payload.

12.78.3.31 CFE_TIME_TimeCmd_t

```
typedef struct CFE_TIME_TimeCmd CFE_TIME_TimeCmd_t
```

Generic seconds, microseconds argument command.

12.78.3.32 CFE_TIME_ToneDataCmd_Payload_t

```
typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t
```

Time at tone data command payload.

12.78.3.33 CFE_TIME_ToneDataCmd_t

```
typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t
```

Time at tone data command.

12.78.3.34 CFE_TIME_ToneSignalCmd_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ToneSignalCmd_t
```

Definition at line 744 of file cfe_time_msg.h.

12.79 osal/docs/src/osal_frontpage.dox File Reference

12.80 osal/docs/src/osal_fs.dox File Reference

12.81 osal/docs/src/osal_timer.dox File Reference

12.82 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)

12.82.1 Detailed Description

Purpose: Unit specification for common types.

Design Notes: Assumes make file has defined processor family

12.82.2 Macro Definition Documentation

12.82.2.1 _EXTENSION_

```
#define _EXTENSION_
```

Definition at line 65 of file common_types.h.

12.82.2.2 CompileTimeAssert

```
#define CompileTimeAssert(
              Condition,
              Message ) typedef char Message[(Condition) ?  1 :  -1]
```

Definition at line 48 of file common_types.h.

12.82.2.3 OS_PRINTF

```
#define OS_PRINTF(
              n,
              m )
```

Definition at line 67 of file common_types.h.

12.82.2.4 OS_USED

```
#define OS_USED
```

Definition at line 66 of file common_types.h.

12.82.2.5 OSAL_BLOCKCOUNT_C

```
#define OSAL_BLOCKCOUNT_C(
              X ) ((osal_blockcount_t)(X))
```

Definition at line 172 of file common_types.h.

12.82.2.6 OSAL_INDEX_C

```
#define OSAL_INDEX_C(
              X ) ((osal_index_t)(X))
```

Definition at line 173 of file common_types.h.

12.82.2.7 OSAL_OBJTYPE_C

```
#define OSAL_OBJTYPE_C(
              X ) ((osal_objtype_t)(X))
```

Definition at line 174 of file common_types.h.

12.82.2.8 OSAL_SIZE_C

```
#define OSAL_SIZE_C(
              X ) ((size_t)(X))
```

Definition at line 171 of file common_types.h.

12.82.2.9 OSAL_STATUS_C

```
#define OSAL_STATUS_C(
              X ) ((osal_status_t)(X))
```

Definition at line 175 of file common_types.h.

12.82.3 Typedef Documentation

12.82.3.1 cpuaddr

```
typedef uintptr_t cpuaddr
```

Definition at line 88 of file common_types.h.

12.82.3.2 cpudiff

```
typedef ptrdiff_t cpudiff
```

Definition at line 90 of file common_types.h.

12.82.3.3 cpusize

```
typedef size_t cpusize
```

Definition at line 89 of file common_types.h.

12.82.3.4 int16

```
typedef int16_t int16
```

Definition at line 80 of file common_types.h.

12.82.3.5 int32

```
typedef int32_t int32
```

Definition at line 81 of file common_types.h.

12.82.3.6 int64

```
typedef int64_t int64
```

Definition at line 82 of file common_types.h.

12.82.3.7 int8

```
typedef int8_t int8
```

Definition at line 79 of file common_types.h.

12.82.3.8 intptr

```
typedef intptr_t intptr
```

Definition at line 87 of file common_types.h.

12.82.3.9 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.

12.82.3.10 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.

12.82.3.11 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.

12.82.3.12 osal_index_t

```
typedef uint32 osal_index_t
```

A type used to represent an index into a table structure

This is used when referring directly to a table index as opposed to an object ID. It is primarily intended for internal use, but is also output from public APIs such as OS_ObjectIdToArrayIndex().

Definition at line 126 of file common_types.h.

12.82.3.13 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.

12.82.3.14 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.

12.82.3.15 uint16

```
typedef uint16_t uint16
```

Definition at line 84 of file common_types.h.

12.82.3.16 uint32

```
typedef uint32_t uint32
```

Definition at line 85 of file common_types.h.

12.82.3.17 uint64

```
typedef uint64_t uint64
```

Definition at line 86 of file common_types.h.

12.82.3.18 uint8

```
typedef uint8_t uint8
```

Definition at line 83 of file common_types.h.

12.82.4 Function Documentation

12.82.4.1 CompileTimeAssert() [1/9]

```
CompileTimeAssert (
            sizeof(uint8)  = =1,
            TypeUint8WrongSize  )
```

12.82.4.2 CompileTimeAssert() [2/9]

```
CompileTimeAssert (
            sizeof(uint16)  = =2,
            TypeUint16WrongSize  )
```

12.82.4.3 CompileTimeAssert() [3/9]

```
CompileTimeAssert (
            sizeof(uint32)  = =4,
            TypeUint32WrongSize  )
```

12.82.4.4 CompileTimeAssert() [4/9]

```
CompileTimeAssert (
            sizeof(uint64)  = =8,
            TypeUint64WrongSize  )
```

12.82.4.5 CompileTimeAssert() [5/9]

```
CompileTimeAssert (
            sizeof(int8)  = =1,
            Typeint8WrongSize  )
```

12.82.4.6 CompileTimeAssert() [6/9]

```
CompileTimeAssert (
            sizeof(int16)  = =2,
            Typeint16WrongSize  )
```

12.82.4.7 CompileTimeAssert() [7/9]

```
CompileTimeAssert (
            sizeof(int32)  = =4,
            Typeint32WrongSize  )
```

12.82.4.8 CompileTimeAssert() [8/9]

```
CompileTimeAssert (
            sizeof(int64)  = =8,
            Typeint64WrongSize  )
```

12.82.4.9 CompileTimeAssert() [9/9]

```
CompileTimeAssert (
            sizeof(cpuaddr) >=sizeof(void *) ,
            TypePtrWrongSize  )
```

12.83 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.

12.83.1 Detailed Description

Declarations and prototypes for binary semaphores

12.84 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)

12.84.1 Detailed Description

Declarations and prototypes for OSAL BSP

12.85 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_TI↩ME_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 }

  Multipliers/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.

- 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.

12.85.1 Detailed Description

Declarations and prototypes for osapi-clock module

12.85.2 Enumeration Type Documentation

12.85.2.1 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

OS_TIME_TICK_RESOLUTION_NS	
OS_TIME_TICKS_PER_SECOND	
OS_TIME_TICKS_PER_MSEC	
OS_TIME_TICKS_PER_USEC	

Definition at line 61 of file osapi-clock.h.

12.86 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.

12.86.1 Detailed Description

Declarations and prototypes for general OSAL functions that are not part of a subsystem

12.86.2 Typedef Documentation

12.86.2.1 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	*event*	The event that occurred
in	*object↩ _id*	The associated object_id, or 0 if not associated with an object
in,out	*data*	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.

12.86.3 Enumeration Type Documentation

12.86.3.1 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.

12.87 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.

12.87.1 Detailed Description

General constants for OSAL that are shared across subsystems

12.87.2 Macro Definition Documentation

12.87.2.1 OS_CHECK

```
#define OS_CHECK (0)
```

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

12.87.2.2 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.

12.87.2.3 OS_OBJECT_CREATOR_ANY

```
#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)

Definition at line 46 of file osapi-constants.h.

Referenced by FM_GetFilenameState(), and FM_GetOpenFilesData().

12.87.2.4 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.

Referenced by FM_ChildConcatCmd(), FM_ChildDeleteAllCmd(), FM_ChildDeleteDirCmd(), FM_ChildDirListFileCmd(), FM_ChildDirListFileInit(), FM_ChildDirListPktCmd(), FM_ChildFileInfoCmd(), and FM_ChildTask().

12.87.2.5 OS_PEND

```
#define OS_PEND (-1)
```

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

12.88 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.

12.88.1 Detailed Description

Declarations and prototypes for counting semaphores

12.89 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_DIRENTRY_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.

12.89.1 Detailed Description

Declarations and prototypes for directories

12.89.2 Macro Definition Documentation

12.89.2.1 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.

12.90 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.

12.90.1 Detailed Description

OSAL error code definitions

12.90.2 Macro Definition Documentation

12.90.2.1 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.

12.90.3 Typedef Documentation

12.90.3.1 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.

12.91 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_T↩RUNCATE = 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.

12.91.1 Detailed Description

Declarations and prototypes for file objects

12.91.2 Macro Definition Documentation

12.91.2.1 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.

12.91.2.2 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.

Referenced by FM_GetFilenameState().

12.91.2.3 OS_FILESTAT_MODE

```
#define OS_FILESTAT_MODE(
            x ) ((x).FileModeBits)
```

Access file stat mode bits.

Definition at line 88 of file osapi-file.h.

Referenced by FM_ChildSizeTimeMode(), and FM_GetFilenameState().

12.91.2.4 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.

12.91.2.5 OS_FILESTAT_SIZE

```
#define OS_FILESTAT_SIZE(
              x ) ((x).FileSize)
```

Access file stat size field.

Definition at line 98 of file osapi-file.h.

Referenced by FM_ChildSizeTimeMode(), and FM_GetFilenameState().

12.91.2.6 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.

Referenced by FM_ChildSizeTimeMode(), and FM_GetFilenameState().

12.91.2.7 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.

12.91.3 Enumeration Type Documentation

12.91.3.1 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.

12.91.3.2 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.

12.92 osal/src/os/inc/osapi-filesys.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- struct os_fsinfo_t

  OSAL file system info.
- struct OS_statvfs_t

Macros

- #define OS_CHK_ONLY 0
- #define OS_REPAIR 1

Functions

- int32 OS_FileSysAddFixedMap (osal_id_t *filesys_id, const char *phys_path, const char *virt_path)

  Create a fixed mapping between an existing directory and a virtual OSAL mount point.
- int32 OS_mkfs (char *address, const char *devname, const char *volname, size_t blocksize, osal_blockcount_t numblocks)

  Makes a file system on the target.
- int32 OS_mount (const char *devname, const char *mountpoint)

  Mounts a file system.
- int32 OS_initfs (char *address, const char *devname, const char *volname, size_t blocksize, osal_blockcount_t numblocks)

  Initializes an existing file system.
- int32 OS_rmfs (const char *devname)

  Removes a file system.
- int32 OS_unmount (const char *mountpoint)

  Unmounts a mounted file system.
- int32 OS_FileSysStatVolume (const char *name, OS_statvfs_t *statbuf)

  Obtains information about size and free space in a volume.
- int32 OS_chkfs (const char *name, bool repair)

  Checks the health of a file system and repairs it if necessary.
- int32 OS_FS_GetPhysDriveName (char *PhysDriveName, const char *MountPoint)

  Obtains the physical drive name associated with a mount point.
- int32 OS_TranslatePath (const char *VirtualPath, char *LocalPath)

  Translates an OSAL Virtual file system path to a host Local path.
- int32 OS_GetFsInfo (os_fsinfo_t *filesys_info)

  Returns information about the file system.

12.92.1 Detailed Description

Declarations and prototypes for file systems

12.92.2 Macro Definition Documentation

12.92.2.1 OS_CHK_ONLY

```
#define OS_CHK_ONLY 0
```

Unused, API takes bool

Definition at line 31 of file osapi-filesys.h.

12.92.2.2 OS_REPAIR

```
#define OS_REPAIR 1
```

Unused, API takes bool

Definition at line 32 of file osapi-filesys.h.

12.93 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.

12.93.1 Detailed Description

Declarations and prototypes for heap functions

12.94 osal/src/os/inc/osapi-idmap.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

Macros

- #define OS_OBJECT_INDEX_MASK 0xFFFF

  Object index mask.
- #define OS_OBJECT_TYPE_SHIFT 16

  Object type shift.
- #define OS_OBJECT_TYPE_UNDEFINED 0x00

  Object type undefined.
- #define OS_OBJECT_TYPE_OS_TASK 0x01

  Object task type.
- #define OS_OBJECT_TYPE_OS_QUEUE 0x02

  Object queue type.
- #define OS_OBJECT_TYPE_OS_COUNTSEM 0x03

  Object counting semaphore type.
- #define OS_OBJECT_TYPE_OS_BINSEM 0x04

  Object binary semaphore type.
- #define OS_OBJECT_TYPE_OS_MUTEX 0x05

  Object mutex type.
- #define OS_OBJECT_TYPE_OS_STREAM 0x06

  Object stream type.
- #define OS_OBJECT_TYPE_OS_DIR 0x07

  Object directory type.
- #define OS_OBJECT_TYPE_OS_TIMEBASE 0x08

  Object timebase type.
- #define OS_OBJECT_TYPE_OS_TIMECB 0x09

  Object timer callback type.
- #define OS_OBJECT_TYPE_OS_MODULE 0x0A

  Object module type.
- #define OS_OBJECT_TYPE_OS_FILESYS 0x0B

  Object file system type.
- #define OS_OBJECT_TYPE_OS_CONSOLE 0x0C

  Object console type.
- #define OS_OBJECT_TYPE_USER 0x10

  Object user type.

Functions

- static unsigned long OS_ObjectIdToInteger (osal_id_t object_id)

  Obtain an integer value corresponding to an object ID.
- static osal_id_t OS_ObjectIdFromInteger (unsigned long value)

  Obtain an osal ID corresponding to an integer value.
- static bool OS_ObjectIdEqual (osal_id_t object_id1, osal_id_t object_id2)

  Check two OSAL object ID values for equality.
- static bool OS_ObjectIdDefined (osal_id_t object_id)

  Check if an object ID is defined.
- int32 OS_GetResourceName (osal_id_t object_id, char *buffer, size_t buffer_size)

  Obtain the name of an object given an arbitrary object ID.

- osal_objtype_t OS_IdentifyObject (osal_id_t object_id)

  Obtain the type of an object given an arbitrary object ID.
- int32 OS_ConvertToArrayIndex (osal_id_t object_id, osal_index_t *ArrayIndex)

  Converts an abstract ID into a number suitable for use as an array index.
- int32 OS_ObjectIdToArrayIndex (osal_objtype_t idtype, osal_id_t object_id, osal_index_t *ArrayIndex)

  Converts an abstract ID into a number suitable for use as an array index.
- void OS_ForEachObject (osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)

  call the supplied callback function for all valid object IDs
- void OS_ForEachObjectOfType (osal_objtype_t objtype, osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)

  call the supplied callback function for valid object IDs of a specific type

12.94.1 Detailed Description

Declarations and prototypes for object IDs

12.94.2 Macro Definition Documentation

12.94.2.1 OS_OBJECT_INDEX_MASK

```
#define OS_OBJECT_INDEX_MASK 0xFFFF
```

Object index mask.

Definition at line 32 of file osapi-idmap.h.

12.94.2.2 OS_OBJECT_TYPE_SHIFT

```
#define OS_OBJECT_TYPE_SHIFT 16
```

Object type shift.

Definition at line 33 of file osapi-idmap.h.

12.95 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.

12.95.1 Detailed Description

Macro definitions that are used across all OSAL subsystems

12.95.2 Macro Definition Documentation

12.95.2.1 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.

12.95.2.2 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.

12.95.2.3 BUGREPORT

```
#define BUGREPORT(
            ...  ) OS_printf(__VA_ARGS__)
```

Definition at line 79 of file osapi-macros.h.

12.95.2.4 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.

12.96 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.

12.96.1 Detailed Description

Declarations and prototypes for module subsystem

12.96.2 Macro Definition Documentation

12.96.2.1 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.

12.96.2.2 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.

12.97 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.

12.97.1 Detailed Description

Declarations and prototypes for mutexes

12.98 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.

12.98.1 Detailed Description

Declarations and prototypes for network subsystem

12.99 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.

12.99.1 Detailed Description

Declarations and prototypes for printf/console output

12.100 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.

12.100.1 Detailed Description

Declarations and prototypes for queue subsystem

12.101 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.

12.101.1 Detailed Description

Declarations and prototypes for select abstraction

12.101.2 Enumeration Type Documentation

12.101.2.1 OS_StreamState_t

```
enum OS_StreamState_t
```

For the OS_SelectSingle() function's in/out StateFlags parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.

See also

OS_SelectSingle()

Enumerator

OS_STREAM_STATE_BOUND	whether the stream is bound
OS_STREAM_STATE_CONNECTED	whether the stream is connected
OS_STREAM_STATE_READABLE	whether the stream is readable
OS_STREAM_STATE_WRITABLE	whether the stream is writable

Definition at line 55 of file osapi-select.h.

12.102 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.

12.102.1 Detailed Description

Declarations and prototypes for shell abstraction

12.103 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_IN↩ET6, 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_SHU↩T_READ = 1, OS_SocketShutdownMode_SHUT_WRITE = 2, OS_SocketShutdownMode_SHUT_READWRITE = 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 *Remote↩ Addr)

  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.

12.103.1 Detailed Description

Declarations and prototypes for sockets abstraction

12.103.2 Macro Definition Documentation

12.103.2.1 OS_SOCKADDR_MAX_LEN

```
#define OS_SOCKADDR_MAX_LEN 28
```

Definition at line 45 of file osapi-sockets.h.

12.103.3 Enumeration Type Documentation

12.103.3.1 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.

12.103.3.2 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_READWRITE	Disable future reading or writing.

Definition at line 79 of file osapi-sockets.h.

12.103.3.3 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.

12.104 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.

12.104.1 Detailed Description

Declarations and prototypes for task abstraction

12.104.2 Macro Definition Documentation

12.104.2.1 OS_FP_ENABLED

```
#define OS_FP_ENABLED 1
```

Floating point enabled state for a task.

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

12.104.2.2 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.

12.104.2.3 OSAL_PRIORITY_C

```
#define OSAL_PRIORITY_C(
              X ) ((osal_priority_t) {X})
```

Definition at line 46 of file osapi-task.h.

12.104.2.4 OSAL_STACKPTR_C

```
#define OSAL_STACKPTR_C(
              X ) ((osal_stackptr_t) {X})
```

Definition at line 53 of file osapi-task.h.

12.104.2.5 OSAL_TASK_STACK_ALLOCATE

```
#define OSAL_TASK_STACK_ALLOCATE OSAL_STACKPTR_C(NULL)
```

Definition at line 54 of file osapi-task.h.

12.104.3 Typedef Documentation

12.104.3.1 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.

12.104.3.2 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.

12.104.3.3 osal_task

```
typedef void osal_task
```

For task entry point.

Definition at line 68 of file osapi-task.h.

12.104.4 Function Documentation

12.104.4.1 osal_task()

```
typedef osal_task (
            (*)(void) osal_task_entry )
```

For task entry point.

12.105 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.

12.105.1 Detailed Description

Declarations and prototypes for timebase abstraction

12.105.2 Typedef Documentation

12.105.2.1 OS_TimerSync_t

```
typedef uint32(* OS_TimerSync_t) (osal_id_t timer_id)
```

Timer sync.

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

12.106 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_Timer↩Callback_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.

12.106.1 Detailed Description

Declarations and prototypes for timer abstraction (app callbacks)

12.106.2 Typedef Documentation

12.106.2.1 OS_TimerCallback_t

```
typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)
```

Timer callback.

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

12.107 osal/src/os/inc/osapi-version.h File Reference

```
#include "common_types.h"
```

Macros

- #define OS_BUILD_NUMBER 66
- #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_RE↩VISION)

  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.

12.107.1 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.

12.107.2 Macro Definition Documentation

12.107.2.1 OS_BUILD_BASELINE

```
#define OS_BUILD_BASELINE "v6.0.0-rc4"
```

Definition at line 38 of file osapi-version.h.

12.107.2.2 OS_BUILD_NUMBER

```
#define OS_BUILD_NUMBER 66
```

Definition at line 37 of file osapi-version.h.

12.107.2.3 OS_MAJOR_VERSION

```
#define OS_MAJOR_VERSION 5
```

Major version number.

Definition at line 43 of file osapi-version.h.

12.107.2.4 OS_MINOR_VERSION

```
#define OS_MINOR_VERSION 0
```

Minor version number.

Definition at line 44 of file osapi-version.h.

12.107.2.5 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.

12.107.2.6 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.

12.107.2.7 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.

12.107.2.8 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.

12.107.2.9 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.

12.107.2.10 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.

12.107.2.11 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.

12.107.2.12 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.

12.107.3 Function Documentation

12.107.3.1 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

12.107.3.2 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.

12.107.3.3 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	*VersionNumbers*	A fixed-size array to be filled with the version numbers

12.107.3.4 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.

12.108 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-filesys.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"
```

12.108.1 Detailed Description

Purpose: Contains functions prototype definitions and variables declarations for the OS Abstraction Layer, Core OS module

12.109 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

  Reset caused by power having been removed and restored.
- #define CFE_PSP_RST_SUBTYPE_PUSH_BUTTON 2

  Reset caused by reset button on the board.
- #define CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND 3

  Reset was caused by a reset line having been stimulated by a hardware special command.
- #define CFE_PSP_RST_SUBTYPE_HW_WATCHDOG 4

  Reset was caused by a watchdog timer expiring.
- #define CFE_PSP_RST_SUBTYPE_RESET_COMMAND 5

  Reset was caused by cFE ES processing a Reset Command .
- #define CFE_PSP_RST_SUBTYPE_EXCEPTION 6

  Reset was caused by a Processor Exception.
- #define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7

  Reset was caused in an unknown manner.
- #define CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET 8

  Reset was caused by a JTAG or BDM connection.
- #define CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET 9

  Reset reverted to a cFE POWERON due to a boot bank switch.
- #define CFE_PSP_RST_SUBTYPE_MAX 10

  Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Functions

- void CFE_PSP_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.

12.109.1 Macro Definition Documentation

12.109.1.1 BUFF_SIZE

```
#define BUFF_SIZE 256
```

Definition at line 80 of file cfe_psp.h.

12.109.1.2 CFE_PSP_ERROR

```
#define CFE_PSP_ERROR (-1)
```

Definition at line 50 of file cfe_psp.h.

12.109.1.3 CFE_PSP_ERROR_ADDRESS_MISALIGNED

```
#define CFE_PSP_ERROR_ADDRESS_MISALIGNED (-3)
```

Definition at line 52 of file cfe_psp.h.

12.109.1.4 CFE_PSP_ERROR_NOT_IMPLEMENTED

```
#define CFE_PSP_ERROR_NOT_IMPLEMENTED (-27)
```

Definition at line 61 of file cfe_psp.h.

12.109.1.5 CFE_PSP_ERROR_TIMEOUT

```
#define CFE_PSP_ERROR_TIMEOUT (-4)
```

Definition at line 53 of file cfe_psp.h.

12.109.1.6 CFE_PSP_INVALID_INT_NUM

```
#define CFE_PSP_INVALID_INT_NUM (-5)
```

Definition at line 54 of file cfe_psp.h.

12.109.1.7 CFE_PSP_INVALID_MEM_ADDR

```
#define CFE_PSP_INVALID_MEM_ADDR (-21)
```

Definition at line 55 of file cfe_psp.h.

12.109.1.8 CFE_PSP_INVALID_MEM_ATTR

```
#define CFE_PSP_INVALID_MEM_ATTR (-26)
```

Definition at line 60 of file cfe_psp.h.

12.109.1.9 CFE_PSP_INVALID_MEM_RANGE

```
#define CFE_PSP_INVALID_MEM_RANGE (-23)
```

Definition at line 57 of file cfe_psp.h.

12.109.1.10 CFE_PSP_INVALID_MEM_SIZE

```
#define CFE_PSP_INVALID_MEM_SIZE (-25)
```

Definition at line 59 of file cfe_psp.h.

12.109.1.11 CFE_PSP_INVALID_MEM_TYPE

```
#define CFE_PSP_INVALID_MEM_TYPE (-22)
```

Definition at line 56 of file cfe_psp.h.

12.109.1.12 CFE_PSP_INVALID_MEM_WORDSIZE

```
#define CFE_PSP_INVALID_MEM_WORDSIZE (-24)
```

Definition at line 58 of file cfe_psp.h.

12.109.1.13 CFE_PSP_INVALID_MODULE_ID

```
#define CFE_PSP_INVALID_MODULE_ID (-29)
```

Definition at line 63 of file cfe_psp.h.

12.109.1.14 CFE_PSP_INVALID_MODULE_NAME

```
#define CFE_PSP_INVALID_MODULE_NAME (-28)
```

Definition at line 62 of file cfe_psp.h.

12.109.1.15 CFE_PSP_INVALID_POINTER

```
#define CFE_PSP_INVALID_POINTER (-2)
```

Definition at line 51 of file cfe_psp.h.

12.109.1.16 CFE_PSP_MEM_ANY

```
#define CFE_PSP_MEM_ANY 3
```

Definition at line 90 of file cfe_psp.h.

12.109.1.17 CFE_PSP_MEM_ATTR_READ

```
#define CFE_PSP_MEM_ATTR_READ 0x02
```

Definition at line 97 of file cfe_psp.h.

12.109.1.18 CFE_PSP_MEM_ATTR_READWRITE

```
#define CFE_PSP_MEM_ATTR_READWRITE 0x03
```

Definition at line 98 of file cfe_psp.h.

12.109.1.19 CFE_PSP_MEM_ATTR_WRITE

```
#define CFE_PSP_MEM_ATTR_WRITE 0x01
```

Definition at line 96 of file cfe_psp.h.

12.109.1.20 CFE_PSP_MEM_EEPROM

```
#define CFE_PSP_MEM_EEPROM 2
```

Definition at line 89 of file cfe_psp.h.

12.109.1.21 CFE_PSP_MEM_INVALID

```
#define CFE_PSP_MEM_INVALID 4
```

Definition at line 91 of file cfe_psp.h.

12.109.1.22 CFE_PSP_MEM_RAM

```
#define CFE_PSP_MEM_RAM 1
```

Definition at line 88 of file cfe_psp.h.

12.109.1.23 CFE_PSP_MEM_SIZE_BYTE

```
#define CFE_PSP_MEM_SIZE_BYTE 0x01
```

Definition at line 103 of file cfe_psp.h.

12.109.1.24 CFE_PSP_MEM_SIZE_DWORD

```
#define CFE_PSP_MEM_SIZE_DWORD 0x04
```

Definition at line 105 of file cfe_psp.h.

12.109.1.25 CFE_PSP_MEM_SIZE_WORD

```
#define CFE_PSP_MEM_SIZE_WORD 0x02
```

Definition at line 104 of file cfe_psp.h.

12.109.1.26 CFE_PSP_NO_EXCEPTION_DATA

```
#define CFE_PSP_NO_EXCEPTION_DATA (-30)
```

Definition at line 64 of file cfe_psp.h.

12.109.1.27 CFE_PSP_PANIC_CORE_APP

```
#define CFE_PSP_PANIC_CORE_APP 6
```

Definition at line 74 of file cfe_psp.h.

12.109.1.28 CFE_PSP_PANIC_GENERAL_FAILURE

```
#define CFE_PSP_PANIC_GENERAL_FAILURE 7
```

Definition at line 75 of file cfe_psp.h.

12.109.1.29 CFE_PSP_PANIC_MEMORY_ALLOC

```
#define CFE_PSP_PANIC_MEMORY_ALLOC 3
```

Definition at line 71 of file cfe_psp.h.

12.109.1.30 CFE_PSP_PANIC_NONVOL_DISK

```
#define CFE_PSP_PANIC_NONVOL_DISK 4
```

Definition at line 72 of file cfe_psp.h.

12.109.1.31 CFE_PSP_PANIC_STARTUP

```
#define CFE_PSP_PANIC_STARTUP 1
```

Definition at line 69 of file cfe_psp.h.

12.109.1.32 CFE_PSP_PANIC_STARTUP_SEM

```
#define CFE_PSP_PANIC_STARTUP_SEM 5
```

Definition at line 73 of file cfe_psp.h.

12.109.1.33 CFE_PSP_PANIC_VOLATILE_DISK

```
#define CFE_PSP_PANIC_VOLATILE_DISK 2
```

Definition at line 70 of file cfe_psp.h.

12.109.1.34 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.

12.109.1.35 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.

12.109.1.36 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.

12.109.1.37 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.

12.109.1.38 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.

12.109.1.39 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.

12.109.1.40 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.

12.109.1.41 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.

12.109.1.42 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.

12.109.1.43 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.

12.109.1.44 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.

12.109.1.45 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.

12.109.1.46 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.

12.109.1.47 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.

12.109.1.48 CFE_PSP_SUCCESS

```
#define CFE_PSP_SUCCESS (0)
```

Definition at line 49 of file cfe_psp.h.

12.109.1.49 SIZE_BYTE

```
#define SIZE_BYTE 1
```

Definition at line 81 of file cfe_psp.h.

12.109.1.50 SIZE_HALF

```
#define SIZE_HALF 2
```

Definition at line 82 of file cfe_psp.h.

12.109.1.51 SIZE_WORD

```
#define SIZE_WORD 3
```

Definition at line 83 of file cfe_psp.h.

12.109.2 Function Documentation

12.109.2.1 CFE_PSP_AttachExceptions()

```
void CFE_PSP_AttachExceptions (
            void  )
```

12.109.2.2 CFE_PSP_Decompress()

```
int32 CFE_PSP_Decompress (
            char * srcFileName,
            char * dstFileName )
```

12.109.2.3 CFE_PSP_EepromPowerDown()

```
int32 CFE_PSP_EepromPowerDown (
            uint32 Bank )
```

12.109.2.4 CFE_PSP_EepromPowerUp()

```
int32 CFE_PSP_EepromPowerUp (
            uint32 Bank )
```

12.109.2.5 CFE_PSP_EepromWrite16()

```
int32 CFE_PSP_EepromWrite16 (
            cpuaddr MemoryAddress,
            uint16 uint16Value )
```

12.109.2.6 CFE_PSP_EepromWrite32()

```
int32 CFE_PSP_EepromWrite32 (
            cpuaddr MemoryAddress,
            uint32 uint32Value )
```

12.109.2.7 CFE_PSP_EepromWrite8()

```
int32 CFE_PSP_EepromWrite8 (
            cpuaddr MemoryAddress,
            uint8 ByteValue )
```

12.109.2.8 CFE_PSP_EepromWriteDisable()

```
int32 CFE_PSP_EepromWriteDisable (
            uint32 Bank )
```

12.109.2.9 CFE_PSP_EepromWriteEnable()

```
int32 CFE_PSP_EepromWriteEnable (
            uint32 Bank )
```

12.109.2.10 CFE_PSP_Exception_CopyContext()

```
int32 CFE_PSP_Exception_CopyContext (
            uint32 ContextLogId,
            void * ContextBuf,
            uint32 ContextSize )
```

12.109.2.11 CFE_PSP_Exception_GetCount()

```
uint32 CFE_PSP_Exception_GetCount (
            void  )
```

12.109.2.12 CFE_PSP_Exception_GetSummary()

```
int32 CFE_PSP_Exception_GetSummary (
            uint32 * ContextLogId,
            osal_id_t * TaskId,
            char * ReasonBuf,
            uint32 ReasonSize )
```

12.109.2.13 CFE_PSP_FlushCaches()

```
void CFE_PSP_FlushCaches (
            uint32 type,
            void * address,
            uint32 size )
```

12.109.2.14 CFE_PSP_Get_Dec()

```
uint32 CFE_PSP_Get_Dec (
            void  )
```

12.109.2.15 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	*Tbu*	Buffer to hold the upper 32 bits of a 64-bit tick counter
out	*Tbl*	Buffer to hold the lower 32 bits of a 64-bit tick counter

12.109.2.16 CFE_PSP_Get_Timer_Tick()

```
uint32 CFE_PSP_Get_Timer_Tick (
            void  )
```

12.109.2.17 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

12.109.2.18 CFE_PSP_GetCDSSize()

```
int32 CFE_PSP_GetCDSSize (
            uint32 * SizeOfCDS )
```

12.109.2.19 CFE_PSP_GetCFETextSegmentInfo()

```
int32 CFE_PSP_GetCFETextSegmentInfo (
            cpuaddr * PtrToCFESegment,
            uint32 * SizeOfCFESegment )
```

12.109.2.20 CFE_PSP_GetKernelTextSegmentInfo()

```
int32 CFE_PSP_GetKernelTextSegmentInfo (
            cpuaddr * PtrToKernelSegment,
            uint32 * SizeOfKernelSegment )
```

12.109.2.21 CFE_PSP_GetProcessorId()

```
uint32 CFE_PSP_GetProcessorId (
            void  )
```

12.109.2.22 CFE_PSP_GetProcessorName()

```
const char* CFE_PSP_GetProcessorName (
            void  )
```

12.109.2.23 CFE_PSP_GetResetArea()

```
int32 CFE_PSP_GetResetArea (
            cpuaddr * PtrToResetArea,
            uint32 * SizeOfResetArea )
```

12.109.2.24 CFE_PSP_GetRestartType()

```
uint32 CFE_PSP_GetRestartType (
            uint32 * restartSubType )
```

12.109.2.25 CFE_PSP_GetSpacecraftId()

```
uint32 CFE_PSP_GetSpacecraftId (
            void  )
```

12.109.2.26 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

out	*LocalTime*	Value of PSP tick counter as OS_time_t

12.109.2.27 CFE_PSP_GetTimerLow32Rollover()

```
uint32 CFE_PSP_GetTimerLow32Rollover (
            void  )
```

12.109.2.28 CFE_PSP_GetTimerTicksPerSecond()

```
uint32 CFE_PSP_GetTimerTicksPerSecond (
            void  )
```

12.109.2.29 CFE_PSP_GetUserReservedArea()

```
int32 CFE_PSP_GetUserReservedArea (
            cpuaddr * PtrToUserArea,
            uint32 * SizeOfUserArea )
```

12.109.2.30 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.

12.109.2.31 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

out	*VersionNumbers*	A fixed-size array to be filled with the version numbers

12.109.2.32 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.

12.109.2.33 CFE_PSP_GetVolatileDiskMem()

```
int32 CFE_PSP_GetVolatileDiskMem (
            cpuaddr * PtrToVolDisk,
            uint32 * SizeOfVolDisk )
```

12.109.2.34 CFE_PSP_InitSSR()

```
int32 CFE_PSP_InitSSR (
            uint32 bus,
            uint32 device,
            char * DeviceName )
```

12.109.2.35 CFE_PSP_Main()

```
void CFE_PSP_Main (
            void  )
```

12.109.2.36 CFE_PSP_MemCpy()

```
int32 CFE_PSP_MemCpy (
            void * dest,
            const void * src,
            uint32 n )
```

Referenced by FM_GetDirListFileCmd().

12.109.2.37 CFE_PSP_MemRangeGet()

```
int32 CFE_PSP_MemRangeGet (
            uint32 RangeNum,
            uint32 * MemoryType,
            cpuaddr * StartAddr,
            size_t * Size,
            size_t * WordSize,
            uint32 * Attributes )
```

12.109.2.38 CFE_PSP_MemRanges()

```
uint32 CFE_PSP_MemRanges (
            void  )
```

12.109.2.39 CFE_PSP_MemRangeSet()

```
int32 CFE_PSP_MemRangeSet (
            uint32 RangeNum,
            uint32 MemoryType,
            cpuaddr StartAddr,
            size_t Size,
            size_t WordSize,
            uint32 Attributes )
```

12.109.2.40 CFE_PSP_MemRead16()

```
int32 CFE_PSP_MemRead16 (
            cpuaddr MemoryAddress,
            uint16 * uint16Value )
```

12.109.2.41 CFE_PSP_MemRead32()

```
int32 CFE_PSP_MemRead32 (
            cpuaddr MemoryAddress,
            uint32 * uint32Value )
```

12.109.2.42 CFE_PSP_MemRead8()

```
int32 CFE_PSP_MemRead8 (
            cpuaddr MemoryAddress,
            uint8 * ByteValue )
```

12.109.2.43 CFE_PSP_MemSet()

```
int32 CFE_PSP_MemSet (
            void * dest,
            uint8 value,
            uint32 n )
```

Referenced by FM_AppInit(), FM_ChildDirListFileInit(), FM_ChildSizeTimeMode(), and FM_VerifyChildTask().

12.109.2.44 CFE_PSP_MemValidateRange()

```
int32 CFE_PSP_MemValidateRange (
            cpuaddr Address,
            size_t Size,
            uint32 MemoryType )
```

12.109.2.45 CFE_PSP_MemWrite16()

```
int32 CFE_PSP_MemWrite16 (
            cpuaddr MemoryAddress,
            uint16 uint16Value )
```

12.109.2.46 CFE_PSP_MemWrite32()

```
int32 CFE_PSP_MemWrite32 (
            cpuaddr MemoryAddress,
            uint32 uint32Value )
```

12.109.2.47 CFE_PSP_MemWrite8()

```
int32 CFE_PSP_MemWrite8 (
            cpuaddr MemoryAddress,
            uint8 ByteValue )
```

12.109.2.48 CFE_PSP_Panic()

```
void CFE_PSP_Panic (
            int32 ErrorCode )
```

12.109.2.49 CFE_PSP_PortRead16()

```
int32 CFE_PSP_PortRead16 (
            cpuaddr PortAddress,
            uint16 * uint16Value )
```

12.109.2.50 CFE_PSP_PortRead32()

```
int32 CFE_PSP_PortRead32 (
            cpuaddr PortAddress,
            uint32 * uint32Value )
```

12.109.2.51 CFE_PSP_PortRead8()

```
int32 CFE_PSP_PortRead8 (
            cpuaddr PortAddress,
            uint8 * ByteValue )
```

12.109.2.52 CFE_PSP_PortWrite16()

```
int32 CFE_PSP_PortWrite16 (
            cpuaddr PortAddress,
            uint16 uint16Value )
```

12.109.2.53 CFE_PSP_PortWrite32()

```
int32 CFE_PSP_PortWrite32 (
            cpuaddr PortAddress,
            uint32 uint32Value )
```

12.109.2.54 CFE_PSP_PortWrite8()

```
int32 CFE_PSP_PortWrite8 (
            cpuaddr PortAddress,
            uint8 ByteValue )
```

12.109.2.55 CFE_PSP_ReadFromCDS()

```
int32 CFE_PSP_ReadFromCDS (
            void * PtrToDataToRead,
            uint32 CDSOffset,
            uint32 NumBytes )
```

12.109.2.56 CFE_PSP_Restart()

```
void CFE_PSP_Restart (
            uint32 resetType )
```

12.109.2.57 CFE_PSP_SetDefaultExceptionEnvironment()

```
void CFE_PSP_SetDefaultExceptionEnvironment (
            void  )
```

12.109.2.58 CFE_PSP_WatchdogDisable()

```
void CFE_PSP_WatchdogDisable (
            void  )
```

12.109.2.59 CFE_PSP_WatchdogEnable()

```
void CFE_PSP_WatchdogEnable (
            void  )
```

12.109.2.60 CFE_PSP_WatchdogGet()

```
uint32 CFE_PSP_WatchdogGet (
            void  )
```

12.109.2.61 CFE_PSP_WatchdogInit()

```
void CFE_PSP_WatchdogInit (
            void  )
```

12.109.2.62 CFE_PSP_WatchdogService()

```
void CFE_PSP_WatchdogService (
            void  )
```

12.109.2.63 CFE_PSP_WatchdogSet()

```
void CFE_PSP_WatchdogSet (
            uint32 WatchdogValue )
```

12.109.2.64 CFE_PSP_WriteToCDS()

```
int32 CFE_PSP_WriteToCDS (
            const void * PtrToDataToWrite,
            uint32 CDSOffset,
            uint32 NumBytes )
```

Index

EXTENSION
  common_types.h, 1595
__attribute__
  fm_freespace.c, 1144

ARGCHECK
  osapi-macros.h, 1624
accuracy
  OS_timebase_prop_t, 947
  OS_timer_prop_t, 948
ActiveBuffer
  CFE_TBL_HousekeepingTlm_Payload, 813
ActiveBufferAddr
  CFE_TBL_TblRegPacket_Payload, 830
ActiveTableFlag
  CFE_TBL_DumpCmd_Payload, 806
  CFE_TBL_ValidateCmd_Payload, 836
ActualLength
  OS_SockAddr_t, 939
addr
  OS_module_prop_t, 936
AddrData
  OS_SockAddr_t, 939
Address
  OS_static_symbol_record_t, 942
AddressesAreValid
  CFE_ES_AppInfo, 667
AlignPtr
  OS_SockAddrData_t, 940
AlignU32
  OS_SockAddrData_t, 940
AppData
  CFE_EVS_HousekeepingTlm_Payload, 742
AppDataFilename
  CFE_EVS_AppDataCmd_Payload, 727
AppEnableStatus
  CFE_EVS_AppTlmData, 736
AppEntryPoint
  CFE_ES_StartAppCmd_Payload, 719
AppFileName
  CFE_ES_AppReloadCmd_Payload, 675
  CFE_ES_StartAppCmd_Payload, 719
AppID
  CFE_EVS_AppTlmData, 736
AppId
  CFE_ES_TaskInfo, 725
  CFE_SB_PipeInfoEntry, 779
AppInfo
  CFE_ES_OneAppTlm_Payload, 703
AppMessageSentCounter
  CFE_EVS_AppTlmData, 737
AppName
  CFE_ES_TaskInfo, 725
  CFE_EVS_AppNameBitMaskCmd_Payload, 729
  CFE_EVS_AppNameCmd_Payload, 731
  CFE_EVS_AppNameEventIDCmd_Payload, 733
  CFE_EVS_AppNameEventIDMaskCmd_Payload, 735
  CFE_EVS_PacketID, 751
  CFE_SB_PipeInfoEntry, 779
  CFE_SB_RoutingFileEntry, 785
  FM_OpenFilesEntry_t, 917
Application
  CFE_ES_AppNameCmd_Payload, 674
  CFE_ES_AppReloadCmd_Payload, 675
  CFE_ES_SendMemPoolStatsCmd_Payload, 711
  CFE_ES_StartAppCmd_Payload, 719
ApplicationID
  CFE_FS_Header, 762
apps/fm/docs/dox_src/cfs_fm.dox, 949
apps/fm/fsw/mission_inc/fm_perfids.h, 949
apps/fm/fsw/platform_inc/fm_msgids.h, 949
apps/fm/fsw/platform_inc/fm_platform_cfg.h, 950
apps/fm/fsw/src/fm_app.c, 951
apps/fm/fsw/src/fm_app.h, 962
apps/fm/fsw/src/fm_child.c, 972
apps/fm/fsw/src/fm_child.h, 1000
apps/fm/fsw/src/fm_cmd_utils.c, 1028
apps/fm/fsw/src/fm_cmd_utils.h, 1048
apps/fm/fsw/src/fm_cmds.c, 1066
apps/fm/fsw/src/fm_cmds.h, 1090
apps/fm/fsw/src/fm_defs.h, 1114
apps/fm/fsw/src/fm_events.h, 1117
apps/fm/fsw/src/fm_msg.h, 1129
apps/fm/fsw/src/fm_msgdefs.h, 1132
apps/fm/fsw/src/fm_tbl.c, 1133
apps/fm/fsw/src/fm_tbl.h, 1137
apps/fm/fsw/src/fm_verify.h, 1142
apps/fm/fsw/src/fm_version.h, 1143
apps/fm/fsw/tables/fm_freespace.c, 1143
AtToneDelay
  CFE_TIME_DiagnosticTlm_Payload, 840
AtToneLatch
  CFE_TIME_DiagnosticTlm_Payload, 840
AtToneLeapSeconds
  CFE_TIME_DiagnosticTlm_Payload, 840
  CFE_TIME_ToneDataCmd_Payload, 870
AtToneMET
  CFE_TIME_DiagnosticTlm_Payload, 840
  CFE_TIME_ToneDataCmd_Payload, 870
AtToneSTCF
  CFE_TIME_DiagnosticTlm_Payload, 840

CFE_TIME_ToneDataCmd_Payload, 870
AtToneState
CFE_TIME_ToneDataCmd_Payload, 870

BSSAddress
CFE_ES_AppInfo, 667
BSSSize
CFE_ES_AppInfo, 668
BUFF_SIZE
cfe_psp.h, 1649
BUGCHECK
osapi-macros.h, 1624
BUGREPORT
osapi-macros.h, 1625
BitMask
CFE_EVS_AppNameBitMaskCmd_Payload, 729
CFE_EVS_BitMaskCmd_Payload, 740
block_size
OS_statvfs_t, 943
BlockSize
CFE_ES_BlockStats, 676
BlockStats
CFE_ES_MemPoolStats, 698
blocks_free
OS_statvfs_t, 943
BootSource
CFE_ES_HousekeepingTlm_Payload, 687
bss_address
OS_module_address_t, 934
bss_size
OS_module_address_t, 934
Buffer
OS_SockAddrData_t, 940
build/docs/osconfig-example.h, 1144
ByteAlign4
CFE_TBL_TblRegPacket_Payload, 830
ByteAlignPad1
CFE_TBL_HousekeepingTlm_Payload, 813
ByteAlignSpare
CFE_ES_CDSRegDumpRec, 677

CCSDS_ExtendedHeader, 664
Subsystem, 664
SystemId, 664
CCSDS_ExtendedHeader_t
ccsds_hdr.h, 1465
CCSDS_PrimaryHeader, 665
Length, 665
Sequence, 665
StreamId, 665
CCSDS_PrimaryHeader_t
ccsds_hdr.h, 1465
cFE Access Table Content APIs, 485
CFE_TBL_GetAddress, 485
CFE_TBL_GetAddresses, 486
CFE_TBL_ReleaseAddress, 487
CFE_TBL_ReleaseAddresses, 488
cFE Application Behavior APIs, 348
CFE_ES_ExitApp, 348
CFE_ES_IncrementTaskCounter, 349
CFE_ES_RunLoop, 349
CFE_ES_WaitForStartupSync, 351
CFE_ES_WaitForSystemState, 351
cFE Application Control APIs, 345
CFE_ES_DeleteApp, 345
CFE_ES_ReloadApp, 346
CFE_ES_RestartApp, 346
cFE Child Task APIs, 363
CFE_ES_CreateChildTask, 363
CFE_ES_DeleteChildTask, 364
CFE_ES_ExitChildTask, 365
CFE_ES_GetTaskIDByName, 365
CFE_ES_GetTaskName, 366
cFE Clock State Flag Defines, 520
CFE_TIME_FLAG_ADD1HZ, 520
CFE_TIME_FLAG_ADDADJ, 520
CFE_TIME_FLAG_ADDTCL, 521
CFE_TIME_FLAG_CLKSET, 521
CFE_TIME_FLAG_CMDFLY, 521
CFE_TIME_FLAG_FLYING, 521
CFE_TIME_FLAG_GDTONE, 521
CFE_TIME_FLAG_REFERR, 522
CFE_TIME_FLAG_SERVER, 522
CFE_TIME_FLAG_SIGPRI, 522
CFE_TIME_FLAG_SRCINT, 522
CFE_TIME_FLAG_SRVFLY, 522
CFE_TIME_FLAG_UNUSED, 522
cFE Critical Data Store APIs, 372
CFE_ES_CopyToCDS, 372
CFE_ES_GetCDSBlockIDByName, 373
CFE_ES_GetCDSBlockName, 374
CFE_ES_RegisterCDS, 374
CFE_ES_RestoreFromCDS, 376
cFE Entry/Exit APIs, 343
CFE_ES_Main, 343
CFE_ES_ResetCFE, 344
cFE External Time Source APIs, 511
CFE_TIME_ExternalGPS, 511
CFE_TIME_ExternalMET, 512
CFE_TIME_ExternalTime, 512
CFE_TIME_ExternalTone, 513
CFE_TIME_RegisterSynchCallback, 514
CFE_TIME_UnregisterSynchCallback, 514
cFE File Header Management APIs, 405
CFE_FS_InitHeader, 405
CFE_FS_ReadHeader, 405
CFE_FS_SetTimestamp, 406
CFE_FS_WriteHeader, 407
cFE File Utility APIs, 410

CFE_FS_BackgroundFileDumpIsPending, 410
CFE_FS_BackgroundFileDumpRequest, 411
CFE_FS_ExtractFilenameFromPath, 411
CFE_FS_GetDefaultExtension, 412
CFE_FS_GetDefaultMountPoint, 413
CFE_FS_ParseInputFileName, 413
CFE_FS_ParseInputFileNameEx, 414
cFE Generic Counter APIs, 388
CFE_ES_DeleteGenCounter, 388
CFE_ES_GetGenCount, 389
CFE_ES_GetGenCounterIDByName, 390
CFE_ES_GetGenCounterName, 390
CFE_ES_IncrementGenCounter, 391
CFE_ES_RegisterGenCounter, 393
CFE_ES_SetGenCount, 394
cFE Generic Message APIs, 416
CFE_MSG_Init, 416
cFE Get Current Time APIs, 498
CFE_TIME_GetMETseconds, 498
CFE_TIME_GetMETsubsecs, 499
CFE_TIME_GetMET, 498
CFE_TIME_GetTAI, 499
CFE_TIME_GetTime, 500
CFE_TIME_GetUTC, 501
cFE Get Table Information APIs, 490
CFE_TBL_GetInfo, 490
CFE_TBL_GetStatus, 491
CFE_TBL_NotifyByMessage, 492
cFE Get Time Information APIs, 502
CFE_TIME_GetClockInfo, 502
CFE_TIME_GetClockState, 502
CFE_TIME_GetLeapSeconds, 503
CFE_TIME_GetSTCF, 503
cFE Information APIs, 353
CFE_ES_GetAppIDByName, 354
CFE_ES_GetAppID, 353
CFE_ES_GetAppInfo, 355
CFE_ES_GetAppName, 356
CFE_ES_GetLibIDByName, 356
CFE_ES_GetLibInfo, 357
CFE_ES_GetLibName, 358
CFE_ES_GetModuleInfo, 359
CFE_ES_GetResetType, 360
CFE_ES_GetTaskID, 361
CFE_ES_GetTaskInfo, 362
cFE Manage Table Content APIs, 478
CFE_TBL_DumpToBuffer, 478
CFE_TBL_Load, 479
CFE_TBL_Manage, 480
CFE_TBL_Modified, 481
CFE_TBL_Update, 482
CFE_TBL_Validate, 483
cFE Memory Manager APIs, 377
CFE_ES_GetMemPoolStats, 377
CFE_ES_GetPoolBuf, 378
CFE_ES_GetPoolBufInfo, 379
CFE_ES_PoolCreate, 380
CFE_ES_PoolCreateEx, 381
CFE_ES_PoolCreateNoSem, 382
CFE_ES_PoolDelete, 383
CFE_ES_PutPoolBuf, 384
cFE Message Characteristics APIs, 463
CFE_SB_GetUserData, 463
CFE_SB_GetUserDataLength, 464
CFE_SB_MessageStringGet, 464
CFE_SB_MessageStringSet, 465
CFE_SB_SetUserDataLength, 466
CFE_SB_TimeStampMsg, 467
cFE Message Extended Header APIs, 428
CFE_MSG_GetEDSVersion, 428
CFE_MSG_GetEndian, 429
CFE_MSG_GetPlaybackFlag, 429
CFE_MSG_GetSubsystem, 430
CFE_MSG_GetSystem, 431
CFE_MSG_SetEDSVersion, 431
CFE_MSG_SetEndian, 432
CFE_MSG_SetPlaybackFlag, 433
CFE_MSG_SetSubsystem, 433
CFE_MSG_SetSystem, 434
cFE Message ID APIs, 468
CFE_SB_IsValidMsgId, 468
CFE_SB_MsgId_Equal, 468
CFE_SB_MsgIdToValue, 469
CFE_SB_ValueToMsgId, 469
cFE Message Id APIs, 441
CFE_MSG_GetMsgId, 441
CFE_MSG_GetTypeFromMsgId, 442
CFE_MSG_SetMsgId, 442
cFE Message Primary Header APIs, 417
CFE_MSG_GetApId, 417
CFE_MSG_GetHasSecondaryHeader, 418
CFE_MSG_GetHeaderVersion, 419
CFE_MSG_GetNextSequenceCount, 419
CFE_MSG_GetSegmentationFlag, 420
CFE_MSG_GetSequenceCount, 420
CFE_MSG_GetSize, 421
CFE_MSG_GetType, 422
CFE_MSG_SetApId, 422
CFE_MSG_SetHasSecondaryHeader, 424
CFE_MSG_SetHeaderVersion, 424
CFE_MSG_SetSegmentationFlag, 425
CFE_MSG_SetSequenceCount, 426
CFE_MSG_SetSize, 426
CFE_MSG_SetType, 427
cFE Message Secondary Header APIs, 435
CFE_MSG_GenerateChecksum, 435
CFE_MSG_GetFcnCode, 436
CFE_MSG_GetMsgTime, 436

CFE_MSG_SetFcnCode, 437
CFE_MSG_SetMsgTime, 439
CFE_MSG_ValidateChecksum, 440
cFE Message Subscription Control APIs, 451
CFE_SB_Subscribe, 451
CFE_SB_SubscribeEx, 452
CFE_SB_SubscribeLocal, 453
CFE_SB_Unsubscribe, 454
CFE_SB_UnsubscribeLocal, 455
cFE Miscellaneous APIs, 368
CFE_ES_BackgroundWakeup, 368
CFE_ES_CalculateCRC, 368
CFE_ES_ProcessAsyncEvent, 370
CFE_ES_WriteToSysLog, 370
cFE Miscellaneous Time APIs, 516
CFE_TIME_Local1HzISR, 516
CFE_TIME_Print, 516
cFE Performance Monitor APIs, 385
CFE_ES_PerfLogAdd, 386
CFE_ES_PerfLogEntry, 385
CFE_ES_PerfLogExit, 386
cFE Pipe Management APIs, 444
CFE_SB_CreatePipe, 444
CFE_SB_DeletePipe, 445
CFE_SB_GetPipeIdByName, 446
CFE_SB_GetPipeName, 447
CFE_SB_GetPipeOpts, 447
CFE_SB_PipeId_ToIndex, 449
CFE_SB_SetPipeOpts, 450
cFE Registration APIs, 396, 472
CFE_EVS_Register, 396
CFE_TBL_Register, 472
CFE_TBL_Share, 475
CFE_TBL_Unregister, 476
cFE Reset Event Filter APIs, 403
CFE_EVS_ResetAllFilters, 403
CFE_EVS_ResetFilter, 403
cFE Resource ID APIs, 339
CFE_ES_AppID_ToIndex, 339
CFE_ES_CounterID_ToIndex, 340
CFE_ES_LibID_ToIndex, 340
CFE_ES_TaskID_ToIndex, 341
cFE Resource ID base values, 518
cFE Return Code Defines, 302
CFE_ES_APP_CLEANUP_ERR, 307
CFE_ES_BAD_ARGUMENT, 307
CFE_ES_BIN_SEM_DELETE_ERR, 307
CFE_ES_BUFFER_NOT_IN_POOL, 308
CFE_ES_CDS_ACCESS_ERROR, 308
CFE_ES_CDS_ALREADY_EXISTS, 308
CFE_ES_CDS_BLOCK_CRC_ERR, 308
CFE_ES_CDS_INSUFFICIENT_MEMORY, 309
CFE_ES_CDS_INVALID_NAME, 309
CFE_ES_CDS_INVALID_SIZE, 309
CFE_ES_CDS_INVALID, 309
CFE_ES_CDS_OWNER_ACTIVE_ERR, 310
CFE_ES_CDS_WRONG_TYPE_ERR, 310
CFE_ES_COUNT_SEM_DELETE_ERR, 310
CFE_ES_ERR_APP_CREATE, 310
CFE_ES_ERR_APP_REGISTER, 311
CFE_ES_ERR_CHILD_TASK_CREATE, 311
CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_T↩ ASK, 311
CFE_ES_ERR_CHILD_TASK_DELETE, 311
CFE_ES_ERR_CHILD_TASK_REGISTER, 312
CFE_ES_ERR_DUPLICATE_NAME, 312
CFE_ES_ERR_LOAD_LIB, 312
CFE_ES_ERR_MEM_BLOCK_SIZE, 312
CFE_ES_ERR_NAME_NOT_FOUND, 313
CFE_ES_ERR_RESOURCEID_NOT_VALID, 313
CFE_ES_ERR_SYS_LOG_FULL, 313
CFE_ES_ERR_SYS_LOG_TRUNCATED, 313
CFE_ES_FILE_CLOSE_ERR, 314
CFE_ES_FILE_IO_ERR, 314
CFE_ES_LIB_ALREADY_LOADED, 314
CFE_ES_MUT_SEM_DELETE_ERR, 314
CFE_ES_NO_RESOURCE_IDS_AVAILABLE, 315
CFE_ES_NOT_IMPLEMENTED, 315
CFE_ES_OPERATION_TIMED_OUT, 315
CFE_ES_POOL_BLOCK_INVALID, 315
CFE_ES_QUEUE_DELETE_ERR, 316
CFE_ES_RST_ACCESS_ERR, 316
CFE_ES_TASK_DELETE_ERR, 316
CFE_ES_TIMER_DELETE_ERR, 316
CFE_EVS_APP_FILTER_OVERLOAD, 317
CFE_EVS_APP_ILLEGAL_APP_ID, 317
CFE_EVS_APP_NOT_REGISTERED, 317
CFE_EVS_EVT_NOT_REGISTERED, 317
CFE_EVS_FILE_WRITE_ERROR, 318
CFE_EVS_INVALID_PARAMETER, 318
CFE_EVS_NOT_IMPLEMENTED, 318
CFE_EVS_RESET_AREA_POINTER, 318
CFE_EVS_UNKNOWN_FILTER, 319
CFE_FS_BAD_ARGUMENT, 319
CFE_FS_FNAME_TOO_LONG, 319
CFE_FS_INVALID_PATH, 319
CFE_FS_NOT_IMPLEMENTED, 320
CFE_SB_BAD_ARGUMENT, 320
CFE_SB_BUF_ALOC_ERR, 320
CFE_SB_BUFFER_INVALID, 320
CFE_SB_INTERNAL_ERR, 321
CFE_SB_MAX_DESTS_MET, 321
CFE_SB_MAX_MSGS_MET, 321
CFE_SB_MAX_PIPES_MET, 321
CFE_SB_MSG_TOO_BIG, 322
CFE_SB_NO_MESSAGE, 322
CFE_SB_NOT_IMPLEMENTED, 322
CFE_SB_PIPE_CR_ERR, 322

CFE_SB_PIPE_RD_ERR, 323
CFE_SB_TIME_OUT, 323
CFE_SB_WRONG_MSG_TYPE, 323
CFE_STATUS_BAD_COMMAND_CODE, 323
CFE_STATUS_EXTERNAL_RESOURCE_FAIL, 324
CFE_STATUS_NO_COUNTER_INCREMENT, 324
CFE_STATUS_NOT_IMPLEMENTED, 324
CFE_STATUS_REQUEST_ALREADY_PENDING, 324
CFE_STATUS_UNKNOWN_MSG_ID, 325
CFE_STATUS_WRONG_MSG_LENGTH, 325
CFE_SUCCESS, 325
CFE_TBL_BAD_ARGUMENT, 325
CFE_TBL_ERR_ACCESS, 326
CFE_TBL_ERR_BAD_CONTENT_ID, 326
CFE_TBL_ERR_BAD_PROCESSOR_ID, 326
CFE_TBL_ERR_BAD_SPACECRAFT_ID, 326
CFE_TBL_ERR_BAD_SUBTYPE_ID, 327
CFE_TBL_ERR_DUMP_ONLY, 327
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE, 327
CFE_TBL_ERR_DUPLICATE_NOT_OWNED, 327
CFE_TBL_ERR_FILE_FOR_WRONG_TABLE, 328
CFE_TBL_ERR_FILE_SIZE_INCONSISTENT, 328
CFE_TBL_ERR_FILE_TOO_LARGE, 328
CFE_TBL_ERR_FILENAME_TOO_LONG, 328
CFE_TBL_ERR_HANDLES_FULL, 329
CFE_TBL_ERR_ILLEGAL_SRC_TYPE, 329
CFE_TBL_ERR_INVALID_HANDLE, 329
CFE_TBL_ERR_INVALID_NAME, 329
CFE_TBL_ERR_INVALID_OPTIONS, 330
CFE_TBL_ERR_INVALID_SIZE, 330
CFE_TBL_ERR_LOAD_IN_PROGRESS, 330
CFE_TBL_ERR_LOAD_INCOMPLETE, 331
CFE_TBL_ERR_NEVER_LOADED, 331
CFE_TBL_ERR_NO_ACCESS, 331
CFE_TBL_ERR_NO_BUFFER_AVAIL, 331
CFE_TBL_ERR_NO_STD_HEADER, 332
CFE_TBL_ERR_NO_TBL_HEADER, 332
CFE_TBL_ERR_PARTIAL_LOAD, 332
CFE_TBL_ERR_REGISTRY_FULL, 332
CFE_TBL_ERR_SHORT_FILE, 333
CFE_TBL_ERR_UNREGISTERED, 333
CFE_TBL_INFO_DUMP_PENDING, 333
CFE_TBL_INFO_NO_UPDATE_PENDING, 333
CFE_TBL_INFO_NO_VALIDATION_PENDING, 334
CFE_TBL_INFO_RECOVERED_TBL, 334
CFE_TBL_INFO_TABLE_LOCKED, 334
CFE_TBL_INFO_UPDATE_PENDING, 334
CFE_TBL_INFO_UPDATED, 335
CFE_TBL_INFO_VALIDATION_PENDING, 335
CFE_TBL_MESSAGE_ERROR, 335
CFE_TBL_NOT_IMPLEMENTED, 335
CFE_TBL_WARN_DUPLICATE, 336
CFE_TBL_WARN_NOT_CRITICAL, 336
CFE_TBL_WARN_PARTIAL_LOAD, 336
CFE_TBL_WARN_SHORT_FILE, 336
CFE_TIME_BAD_ARGUMENT, 337
CFE_TIME_CALLBACK_NOT_REGISTERED, 337
CFE_TIME_INTERNAL_ONLY, 337
CFE_TIME_NOT_IMPLEMENTED, 337
CFE_TIME_OUT_OF_RANGE, 338
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS, 338
cFE SB Pipe options, 471
CFE_SB_PIPEOPTS_IGNOREMINE, 471
cFE Send Event APIs, 398
CFE_EVS_SendEvent, 398
CFE_EVS_SendEventWithAppID, 399
CFE_EVS_SendTimedEvent, 401
cFE Send/Receive Message APIs, 457
CFE_SB_ReceiveBuffer, 457
CFE_SB_TransmitMsg, 458
cFE Table Type Defines, 494
CFE_TBL_OPT_BUFFER_MSK, 494
CFE_TBL_OPT_CRITICAL_MSK, 495
CFE_TBL_OPT_CRITICAL, 494
CFE_TBL_OPT_DBL_BUFFER, 495
CFE_TBL_OPT_DEFAULT, 495
CFE_TBL_OPT_DUMP_ONLY, 495
CFE_TBL_OPT_LD_DMP_MSK, 495
CFE_TBL_OPT_LOAD_DUMP, 496
CFE_TBL_OPT_NOT_CRITICAL, 496
CFE_TBL_OPT_NOT_USR_DEF, 496
CFE_TBL_OPT_SNGL_BUFFER, 496
CFE_TBL_OPT_USR_DEF_ADDR, 496
CFE_TBL_OPT_USR_DEF_MSK, 497
cFE Time Arithmetic APIs, 505
CFE_TIME_Add, 505
CFE_TIME_Compare, 506
CFE_TIME_Subtract, 507
cFE Time Conversion APIs, 508
CFE_TIME_MET2SCTime, 508
CFE_TIME_Micro2SubSecs, 508
CFE_TIME_Sub2MicroSecs, 509
cFE Zero Copy APIs, 460
CFE_SB_AllocateMessageBuffer, 460
CFE_SB_ReleaseMessageBuffer, 461
CFE_SB_TransmitBuffer, 461
CFE_BIT
cfe_sb.h, 1318
CFE_BUILD_BASELINE
cfe_version.h, 1347
CFE_BUILD_NUMBER
cfe_version.h, 1347
CFE_CLR
cfe_sb.h, 1319
CFE_CONFIGID_UNDEFINED
cfe_config_api_typedefs.h, 1247

CFE_CONFIGID_C
cfe_config_api_typedefs.h, 1247
CFE_Config_Callback_t
cfe_config_api_typedefs.h, 1248
CFE_Config_GetIdByName
cfe_config.h, 1244
CFE_Config_GetName
cfe_config.h, 1244
CFE_Config_GetObjPointer
cfe_config.h, 1245
CFE_Config_GetString
cfe_config.h, 1245
CFE_Config_GetValue
cfe_config.h, 1246
CFE_Config_IterateAll
cfe_config.h, 1246
CFE_ConfigId_t
cfe_config_api_typedefs.h, 1248
CFE_ES_ALL_APPS_EID
cfe_es_events.h, 1352
CFE_ES_APP_CLEANUP_ERR
cFE Return Code Defines, 307
CFE_ES_APP_RESTART
cfe_es_api_typedefs.h, 1264
CFE_ES_APP_TLM_MID
cpu1_msgids.h, 1155
CFE_ES_APPID_UNDEFINED
cfe_es_api_typedefs.h, 1264
CFE_ES_APPID_C
cfe_es_api_typedefs.h, 1264
CFE_ES_AppID_ToIndex
cFE Resource ID APIs, 339
CFE_ES_AppId_t
cfe_es_extern_typedefs.h, 1272
CFE_ES_AppInfo, 666
AddressesAreValid, 667
BSSAddress, 667
BSSSize, 668
CodeAddress, 668
CodeSize, 668
DataAddress, 668
DataSize, 669
EntryPoint, 669
ExceptionAction, 669
ExecutionCounter, 669
FileName, 670
MainTaskId, 670
MainTaskName, 670
Name, 670
NumOfChildTasks, 671
Priority, 671
ResourceId, 671
StackSize, 671
StartAddress, 672
Type, 672
CFE_ES_AppInfo_t
cfe_es_extern_typedefs.h, 1272
CFE_ES_AppNameCmd, 672
CommandHeader, 673
Payload, 673
CFE_ES_AppNameCmd_Payload, 673
Application, 674
CFE_ES_AppNameCmd_Payload_t
cfe_es_msg.h, 1410
CFE_ES_AppNameCmd_t
cfe_es_msg.h, 1410
CFE_ES_AppReloadCmd_Payload, 674
AppFileName, 675
Application, 675
CFE_ES_AppReloadCmd_Payload_t
cfe_es_msg.h, 1410
CFE_ES_AppState
cfe_es_extern_typedefs.h, 1278
CFE_ES_AppState_Enum_t
cfe_es_extern_typedefs.h, 1272
CFE_ES_AppType
cfe_es_extern_typedefs.h, 1278
CFE_ES_AppType_Enum_t
cfe_es_extern_typedefs.h, 1272
CFE_ES_BAD_ARGUMENT
cFE Return Code Defines, 307
CFE_ES_BIN_SEM_DELETE_ERR
cFE Return Code Defines, 307
CFE_ES_BOOT_ERR_EID
cfe_es_events.h, 1352
CFE_ES_BUFFER_NOT_IN_POOL
cFE Return Code Defines, 308
CFE_ES_BUILD_INF_EID
cfe_es_events.h, 1353
CFE_ES_BackgroundWakeup
cFE Miscellaneous APIs, 368
CFE_ES_BlockStats, 675
BlockSize, 676
NumCreated, 676
NumFree, 676
CFE_ES_BlockStats_t
cfe_es_extern_typedefs.h, 1273
CFE_ES_CC1_ERR_EID
cfe_es_events.h, 1353
CFE_ES_CDS_ACCESS_ERROR
cFE Return Code Defines, 308
CFE_ES_CDS_ALREADY_EXISTS
cFE Return Code Defines, 308
CFE_ES_CDS_BAD_HANDLE
cfe_es_api_typedefs.h, 1264
CFE_ES_CDS_BLOCK_CRC_ERR
cFE Return Code Defines, 308
CFE_ES_CDS_DELETE_ERR_EID

cfe_es_events.h, 1354
CFE_ES_CDS_DELETE_TBL_ERR_EID
cfe_es_events.h, 1354
CFE_ES_CDS_DELETED_INFO_EID
cfe_es_events.h, 1354
CFE_ES_CDS_DUMP_ERR_EID
cfe_es_events.h, 1355
CFE_ES_CDS_INSUFFICIENT_MEMORY
cFE Return Code Defines, 309
CFE_ES_CDS_INVALID_NAME
cFE Return Code Defines, 309
CFE_ES_CDS_INVALID_SIZE
cFE Return Code Defines, 309
CFE_ES_CDS_INVALID
cFE Return Code Defines, 309
CFE_ES_CDS_NAME_ERR_EID
cfe_es_events.h, 1355
CFE_ES_CDS_OWNER_ACTIVE_EID
cfe_es_events.h, 1355
CFE_ES_CDS_OWNER_ACTIVE_ERR
cFE Return Code Defines, 310
CFE_ES_CDS_REG_DUMP_INF_EID
cfe_es_events.h, 1356
CFE_ES_CDS_REGISTER_ERR_EID
cfe_es_events.h, 1356
CFE_ES_CDS_WRONG_TYPE_ERR
cFE Return Code Defines, 310
CFE_ES_CDSHANDLE_C
cfe_es_api_typedefs.h, 1264
CFE_ES_CDSHandle_t
cfe_es_extern_typedefs.h, 1273
CFE_ES_CDSRegDumpRec, 676
ByteAlignSpare, 677
Handle, 677
Name, 677
Size, 678
Table, 678
CFE_ES_CDSRegDumpRec_t
cfe_es_extern_typedefs.h, 1273
CFE_ES_CLEAR_ER_LOG_CC
cfe_es_msg.h, 1386
CFE_ES_CLEAR_SYSLOG_CC
cfe_es_msg.h, 1386
CFE_ES_CMD_MID
cpu1_msgids.h, 1155
CFE_ES_COUNT_SEM_DELETE_ERR
cFE Return Code Defines, 310
CFE_ES_COUNTERID_UNDEFINED
cfe_es_api_typedefs.h, 1265
CFE_ES_COUNTERID_C
cfe_es_api_typedefs.h, 1265
CFE_ES_CREATING_CDS_DUMP_ERR_EID
cfe_es_events.h, 1356
CFE_ES_CalculateCRC
cFE Miscellaneous APIs, 368
CFE_ES_ChildTaskMainFuncPtr_t
cfe_es_api_typedefs.h, 1267
CFE_ES_ClearERLogCmd_t
cfe_es_msg.h, 1410
CFE_ES_ClearSysLogCmd_t
cfe_es_msg.h, 1410
CFE_ES_CopyToCDS
cFE Critical Data Store APIs, 372
CFE_ES_CounterID_ToIndex
cFE Resource ID APIs, 340
CFE_ES_CounterId_t
cfe_es_extern_typedefs.h, 1273
CFE_ES_CreateChildTask
cFE Child Task APIs, 363
CFE_ES_DBIT
cfe_es.h, 1261
CFE_ES_DELETE_CDS_CC
cfe_es_msg.h, 1387
CFE_ES_DTEST
cfe_es.h, 1261
CFE_ES_DUMP_CDS_REGISTRY_CC
cfe_es_msg.h, 1388
CFE_ES_DeleteApp
cFE Application Control APIs, 345
CFE_ES_DeleteCDSCmd, 678
CommandHeader, 679
Payload, 679
CFE_ES_DeleteCDSCmd_Payload, 679
CdsName, 680
CFE_ES_DeleteCDSCmd_Payload_t
cfe_es_msg.h, 1410
CFE_ES_DeleteCDSCmd_t
cfe_es_msg.h, 1410
CFE_ES_DeleteChildTask
cFE Child Task APIs, 364
CFE_ES_DeleteGenCounter
cFE Generic Counter APIs, 388
CFE_ES_DumpCDSRegistryCmd, 680
CommandHeader, 681
Payload, 681
CFE_ES_DumpCDSRegistryCmd_Payload, 681
DumpFilename, 682
CFE_ES_DumpCDSRegistryCmd_Payload_t
cfe_es_msg.h, 1411
CFE_ES_DumpCDSRegistryCmd_t
cfe_es_msg.h, 1411
CFE_ES_ERLOG1_INF_EID
cfe_es_events.h, 1357
CFE_ES_ERLOG2_EID
cfe_es_events.h, 1357
CFE_ES_ERLOG2_ERR_EID
cfe_es_events.h, 1357
CFE_ES_ERLOG_PENDING_ERR_EID

cfe_es_events.h, 1358
CFE_ES_ERR_APP_CREATE
cFE Return Code Defines, 310
CFE_ES_ERR_APP_REGISTER
cFE Return Code Defines, 311
CFE_ES_ERR_CHILD_TASK_CREATE
cFE Return Code Defines, 311
CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK
cFE Return Code Defines, 311
CFE_ES_ERR_CHILD_TASK_DELETE
cFE Return Code Defines, 311
CFE_ES_ERR_CHILD_TASK_REGISTER
cFE Return Code Defines, 312
CFE_ES_ERR_DUPLICATE_NAME
cFE Return Code Defines, 312
CFE_ES_ERR_LOAD_LIB
cFE Return Code Defines, 312
CFE_ES_ERR_MEM_BLOCK_SIZE
cFE Return Code Defines, 312
CFE_ES_ERR_NAME_NOT_FOUND
cFE Return Code Defines, 313
CFE_ES_ERR_RESOURCEID_NOT_VALID
cFE Return Code Defines, 313
CFE_ES_ERR_SYS_LOG_FULL
cFE Return Code Defines, 313
CFE_ES_ERR_SYS_LOG_TRUNCATED
cFE Return Code Defines, 313
CFE_ES_ERR_SYSLOGMODE_EID
cfe_es_events.h, 1358
CFE_ES_ERREXIT_APP_ERR_EID
cfe_es_events.h, 1358
CFE_ES_ERREXIT_APP_INF_EID
cfe_es_events.h, 1359
CFE_ES_EXIT_APP_ERR_EID
cfe_es_events.h, 1359
CFE_ES_EXIT_APP_INF_EID
cfe_es_events.h, 1360
CFE_ES_ExceptionAction
cfe_es_extern_typedefs.h, 1279
CFE_ES_ExceptionAction_Enum_t
cfe_es_extern_typedefs.h, 1274
CFE_ES_ExitApp
cFE Application Behavior APIs, 348
CFE_ES_ExitChildTask
cFE Child Task APIs, 365
CFE_ES_FILE_CLOSE_ERR
cFE Return Code Defines, 314
CFE_ES_FILE_IO_ERR
cFE Return Code Defines, 314
CFE_ES_FILEWRITE_ERR_EID
cfe_es_events.h, 1360
CFE_ES_FileNameCmd, 682
CommandHeader, 682
Payload, 682
CFE_ES_FileNameCmd_Payload, 683
FileName, 683
CFE_ES_FileNameCmd_Payload_t
cfe_es_msg.h, 1411
CFE_ES_FileNameCmd_t
cfe_es_msg.h, 1411
CFE_ES_GetAppIDByName
cFE Information APIs, 354
CFE_ES_GetAppID
cFE Information APIs, 353
CFE_ES_GetAppInfo
cFE Information APIs, 355
CFE_ES_GetAppName
cFE Information APIs, 356
CFE_ES_GetCDSBlockIDByName
cFE Critical Data Store APIs, 373
CFE_ES_GetCDSBlockName
cFE Critical Data Store APIs, 374
CFE_ES_GetGenCount
cFE Generic Counter APIs, 389
CFE_ES_GetGenCounterIDByName
cFE Generic Counter APIs, 390
CFE_ES_GetGenCounterName
cFE Generic Counter APIs, 390
CFE_ES_GetLibIDByName
cFE Information APIs, 356
CFE_ES_GetLibInfo
cFE Information APIs, 357
CFE_ES_GetLibName
cFE Information APIs, 358
CFE_ES_GetMemPoolStats
cFE Memory Manager APIs, 377
CFE_ES_GetModuleInfo
cFE Information APIs, 359
CFE_ES_GetPoolBuf
cFE Memory Manager APIs, 378
CFE_ES_GetPoolBufInfo
cFE Memory Manager APIs, 379
CFE_ES_GetResetType
cFE Information APIs, 360
CFE_ES_GetTaskIDByName
cFE Child Task APIs, 365
CFE_ES_GetTaskID
cFE Information APIs, 361
CFE_ES_GetTaskInfo
cFE Information APIs, 362
CFE_ES_GetTaskName
cFE Child Task APIs, 366
CFE_ES_HK_TLM_MID
cpu1_msgids.h, 1155
CFE_ES_HousekeepingTlm, 684
Payload, 684
TelemetryHeader, 684
CFE_ES_HousekeepingTlm_Payload, 684

BootSource, 687
CFECoreChecksum, 687
CFEMajorVersion, 687
CFEMinorVersion, 687
CFEMissionRevision, 687
CFERevision, 688
CommandCounter, 688
CommandErrorCounter, 688
ERLogEntries, 688
ERLogIndex, 689
HeapBlocksFree, 689
HeapBytesFree, 689
HeapMaxBlockSize, 689
MaxProcessorResets, 690
OSALMajorVersion, 690
OSALMinorVersion, 690
OSALMissionRevision, 690
OSALRevision, 691
PSPMajorVersion, 693
PSPMinorVersion, 694
PSPMissionRevision, 694
PSPRevision, 694
PerfDataCount, 691
PerfDataEnd, 691
PerfDataStart, 691
PerfDataToWrite, 692
PerfFilterMask, 692
PerfMode, 692
PerfState, 692
PerfTriggerCount, 693
PerfTriggerMask, 693
ProcessorResets, 693
RegisteredCoreApps, 694
RegisteredExternalApps, 695
RegisteredLibs, 695
RegisteredTasks, 695
ResetSubtype, 695
ResetType, 696
SysLogBytesUsed, 696
SysLogEntries, 696
SysLogMode, 696
SysLogSize, 697
CFE_ES_HousekeepingTlm_Payload_t
cfe_es_msg.h, 1411
CFE_ES_HousekeepingTlm_t
cfe_es_msg.h, 1411
CFE_ES_INIT_INF_EID
cfe_es_events.h, 1360
CFE_ES_INITSTATS_INF_EID
cfe_es_events.h, 1361
CFE_ES_INVALID_POOL_HANDLE_ERR_EID
cfe_es_events.h, 1361
CFE_ES_IncrementGenCounter
cFE Generic Counter APIs, 391
CFE_ES_IncrementTaskCounter
cFE Application Behavior APIs, 349
CFE_ES_LEN_ERR_EID
cfe_es_events.h, 1361
CFE_ES_LIB_ALREADY_LOADED
cFE Return Code Defines, 314
CFE_ES_LIBID_UNDEFINED
cfe_es_api_typedefs.h, 1265
CFE_ES_LIBID_C
cfe_es_api_typedefs.h, 1265
CFE_ES_LibID_ToIndex
cFE Resource ID APIs, 340
CFE_ES_LibId_t
cfe_es_extern_typedefs.h, 1274
CFE_ES_LibraryEntryFuncPtr_t
cfe_es_api_typedefs.h, 1267
CFE_ES_LogEntryType
cfe_es_extern_typedefs.h, 1279
CFE_ES_LogEntryType_Enum_t
cfe_es_extern_typedefs.h, 1274
CFE_ES_LogMode
cfe_es_extern_typedefs.h, 1279
CFE_ES_LogMode_Enum_t
cfe_es_extern_typedefs.h, 1274
CFE_ES_MEMADDRESS_C
cfe_es_extern_typedefs.h, 1271
CFE_ES_MEMHANDLE_UNDEFINED
cfe_es_api_typedefs.h, 1265
CFE_ES_MEMHANDLE_C
cfe_es_api_typedefs.h, 1265
CFE_ES_MEMOFFSET_C
cfe_es_extern_typedefs.h, 1271
CFE_ES_MEMPOOLBUF_C
cfe_es_api_typedefs.h, 1266
CFE_ES_MEMSTATS_TLM_MID
cpu1_msgids.h, 1155
CFE_ES_MID_ERR_EID
cfe_es_events.h, 1362
CFE_ES_MUT_SEM_DELETE_ERR
cFE Return Code Defines, 314
CFE_ES_Main
cFE Entry/Exit APIs, 343
CFE_ES_MemAddress_t
cfe_es_extern_typedefs.h, 1275
CFE_ES_MemHandle_t
cfe_es_extern_typedefs.h, 1275
CFE_ES_MemOffset_t
cfe_es_extern_typedefs.h, 1275
CFE_ES_MemPoolBuf_t
cfe_es_api_typedefs.h, 1268
CFE_ES_MemPoolStats, 697
BlockStats, 698
CheckErrCtr, 698
NumBlocksRequested, 698

NumFreeBytes, 699
PoolSize, 699
CFE_ES_MemPoolStats_t
cfe_es_extern_typedefs.h, 1276
CFE_ES_MemStatsTlm, 699
Payload, 700
TelemetryHeader, 700
CFE_ES_MemStatsTlm_t
cfe_es_msg.h, 1412
CFE_ES_NO_MUTEX
cfe_es_api_typedefs.h, 1266
CFE_ES_NO_RESOURCE_IDS_AVAILABLE
cFE Return Code Defines, 315
CFE_ES_NOOP_CC
cfe_es_msg.h, 1389
CFE_ES_NOOP_INF_EID
cfe_es_events.h, 1362
CFE_ES_NOT_IMPLEMENTED
cFE Return Code Defines, 315
CFE_ES_NoArgsCmd, 700
CommandHeader, 701
CFE_ES_NoArgsCmd_t
cfe_es_msg.h, 1412
CFE_ES_NoopCmd_t
cfe_es_msg.h, 1412
CFE_ES_ONE_APP_EID
cfe_es_events.h, 1362
CFE_ES_ONE_APPID_ERR_EID
cfe_es_events.h, 1363
CFE_ES_ONE_ERR_EID
cfe_es_events.h, 1363
CFE_ES_OPERATION_TIMED_OUT
cFE Return Code Defines, 315
CFE_ES_OSCREATE_ERR_EID
cfe_es_events.h, 1363
CFE_ES_OVER_WRITE_SYSLOG_CC
cfe_es_msg.h, 1390
CFE_ES_OneAppTlm, 701
Payload, 702
TelemetryHeader, 702
CFE_ES_OneAppTlm_Payload, 702
AppInfo, 703
CFE_ES_OneAppTlm_Payload_t
cfe_es_msg.h, 1412
CFE_ES_OneAppTlm_t
cfe_es_msg.h, 1412
CFE_ES_OverWriteSysLogCmd, 703
CommandHeader, 703
Payload, 703
CFE_ES_OverWriteSysLogCmd_Payload, 704
Mode, 704
CFE_ES_OverWriteSysLogCmd_Payload_t
cfe_es_msg.h, 1412
CFE_ES_OverWriteSysLogCmd_t
cfe_es_msg.h, 1413
CFE_ES_PCR_ERR1_EID
cfe_es_events.h, 1364
CFE_ES_PCR_ERR2_EID
cfe_es_events.h, 1364
CFE_ES_PERF_DATAWRITTEN_EID
cfe_es_events.h, 1364
CFE_ES_PERF_FILTMSKCMD_EID
cfe_es_events.h, 1365
CFE_ES_PERF_FILTMSKERR_EID
cfe_es_events.h, 1365
CFE_ES_PERF_LOG_ERR_EID
cfe_es_events.h, 1365
CFE_ES_PERF_STARTCMD_EID
cfe_es_events.h, 1366
CFE_ES_PERF_STARTCMD_ERR_EID
cfe_es_events.h, 1366
CFE_ES_PERF_STARTCMD_TRIG_ERR_EID
cfe_es_events.h, 1366
CFE_ES_PERF_STOPCMD_EID
cfe_es_events.h, 1367
CFE_ES_PERF_STOPCMD_ERR2_EID
cfe_es_events.h, 1367
CFE_ES_PERF_TRIGMSKCMD_EID
cfe_es_events.h, 1367
CFE_ES_PERF_TRIGMSKERR_EID
cfe_es_events.h, 1368
CFE_ES_POOL_BLOCK_INVALID
cFE Return Code Defines, 315
CFE_ES_PerfLogAdd
cFE Performance Monitor APIs, 386
CFE_ES_PerfLogEntry
cFE Performance Monitor APIs, 385
CFE_ES_PerfLogExit
cFE Performance Monitor APIs, 386
CFE_ES_PoolAlign, 705
LongDouble, 705
LongInt, 705
Ptr, 705
CFE_ES_PoolAlign_t
cfe_es_api_typedefs.h, 1268
CFE_ES_PoolCreate
cFE Memory Manager APIs, 380
CFE_ES_PoolCreateEx
cFE Memory Manager APIs, 381
CFE_ES_PoolCreateNoSem
cFE Memory Manager APIs, 382
CFE_ES_PoolDelete
cFE Memory Manager APIs, 383
CFE_ES_PoolStatsTlm_Payload, 706
PoolHandle, 706
PoolStats, 706
CFE_ES_PoolStatsTlm_Payload_t
cfe_es_msg.h, 1413

CFE_ES_ProcessAsyncEvent
cFE Miscellaneous APIs, 370
CFE_ES_PutPoolBuf
cFE Memory Manager APIs, 384
CFE_ES_QUERY_ALL_CC
cfe_es_msg.h, 1391
CFE_ES_QUERY_ALL_TASKS_CC
cfe_es_msg.h, 1392
CFE_ES_QUERY_ONE_CC
cfe_es_msg.h, 1393
CFE_ES_QUEUE_DELETE_ERR
cFE Return Code Defines, 316
CFE_ES_QueryAllCmd_t
cfe_es_msg.h, 1413
CFE_ES_QueryAllTasksCmd_t
cfe_es_msg.h, 1413
CFE_ES_QueryOneCmd_t
cfe_es_msg.h, 1413
CFE_ES_RELOAD_APP_CC
cfe_es_msg.h, 1394
CFE_ES_RELOAD_APP_DBG_EID
cfe_es_events.h, 1368
CFE_ES_RELOAD_APP_ERR1_EID
cfe_es_events.h, 1368
CFE_ES_RELOAD_APP_ERR2_EID
cfe_es_events.h, 1369
CFE_ES_RELOAD_APP_ERR3_EID
cfe_es_events.h, 1369
CFE_ES_RELOAD_APP_ERR4_EID
cfe_es_events.h, 1369
CFE_ES_RELOAD_APP_INF_EID
cfe_es_events.h, 1370
CFE_ES_RESET_COUNTERS_CC
cfe_es_msg.h, 1395
CFE_ES_RESET_INF_EID
cfe_es_events.h, 1370
CFE_ES_RESET_PR_COUNT_CC
cfe_es_msg.h, 1396
CFE_ES_RESET_PR_COUNT_EID
cfe_es_events.h, 1370
CFE_ES_RESTART_APP_CC
cfe_es_msg.h, 1397
CFE_ES_RESTART_APP_DBG_EID
cfe_es_events.h, 1371
CFE_ES_RESTART_APP_ERR1_EID
cfe_es_events.h, 1371
CFE_ES_RESTART_APP_ERR2_EID
cfe_es_events.h, 1371
CFE_ES_RESTART_APP_ERR3_EID
cfe_es_events.h, 1372
CFE_ES_RESTART_APP_ERR4_EID
cfe_es_events.h, 1372
CFE_ES_RESTART_APP_INF_EID
cfe_es_events.h, 1372
CFE_ES_RESTART_CC
cfe_es_msg.h, 1398
CFE_ES_RST_ACCESS_ERR
cFE Return Code Defines, 316
CFE_ES_RegisterCDS
cFE Critical Data Store APIs, 374
CFE_ES_RegisterGenCounter
cFE Generic Counter APIs, 393
CFE_ES_ReloadApp
cFE Application Control APIs, 346
CFE_ES_ReloadAppCmd, 707
CommandHeader, 707
Payload, 707
CFE_ES_ReloadAppCmd_t
cfe_es_msg.h, 1413
CFE_ES_ResetCFE
cFE Entry/Exit APIs, 344
CFE_ES_ResetCountersCmd_t
cfe_es_msg.h, 1414
CFE_ES_ResetPRCountCmd_t
cfe_es_msg.h, 1414
CFE_ES_RestartApp
cFE Application Control APIs, 346
CFE_ES_RestartAppCmd_t
cfe_es_msg.h, 1414
CFE_ES_RestartCmd, 708
CommandHeader, 708
Payload, 708
CFE_ES_RestartCmd_Payload, 709
RestartType, 709
CFE_ES_RestartCmd_Payload_t
cfe_es_msg.h, 1414
CFE_ES_RestartCmd_t
cfe_es_msg.h, 1414
CFE_ES_RestoreFromCDS
cFE Critical Data Store APIs, 376
CFE_ES_RunLoop
cFE Application Behavior APIs, 349
CFE_ES_RunStatus
cfe_es_extern_typedefs.h, 1280
CFE_ES_RunStatus_Enum_t
cfe_es_extern_typedefs.h, 1276
CFE_ES_SEND_HK_MID
cpu1_msgids.h, 1155
CFE_ES_SEND_MEM_POOL_STATS_CC
cfe_es_msg.h, 1399
CFE_ES_SET_MAX_PR_COUNT_CC
cfe_es_msg.h, 1400
CFE_ES_SET_MAX_PR_COUNT_EID
cfe_es_events.h, 1373
CFE_ES_SET_PERF_FILTER_MASK_CC
cfe_es_msg.h, 1401
CFE_ES_SET_PERF_TRIGGER_MASK_CC
cfe_es_msg.h, 1402

CFE_ES_START_APP_CC
cfe_es_msg.h, 1403
CFE_ES_START_ERR_EID
cfe_es_events.h, 1373
CFE_ES_START_EXC_ACTION_ERR_EID
cfe_es_events.h, 1373
CFE_ES_START_INF_EID
cfe_es_events.h, 1374
CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID
cfe_es_events.h, 1374
CFE_ES_START_INVALID_FILENAME_ERR_EID
cfe_es_events.h, 1374
CFE_ES_START_NULL_APP_NAME_ERR_EID
cfe_es_events.h, 1375
CFE_ES_START_PERF_DATA_CC
cfe_es_msg.h, 1404
CFE_ES_START_PRIORITY_ERR_EID
cfe_es_events.h, 1375
CFE_ES_STATIC_POOL_TYPE
cfe_es_api_typedefs.h, 1266
CFE_ES_STOP_APP_CC
cfe_es_msg.h, 1405
CFE_ES_STOP_DBG_EID
cfe_es_events.h, 1375
CFE_ES_STOP_ERR1_EID
cfe_es_events.h, 1376
CFE_ES_STOP_ERR2_EID
cfe_es_events.h, 1376
CFE_ES_STOP_ERR3_EID
cfe_es_events.h, 1376
CFE_ES_STOP_INF_EID
cfe_es_events.h, 1377
CFE_ES_STOP_PERF_DATA_CC
cfe_es_msg.h, 1406
CFE_ES_SYSLOG1_INF_EID
cfe_es_events.h, 1377
CFE_ES_SYSLOG2_EID
cfe_es_events.h, 1377
CFE_ES_SYSLOG2_ERR_EID
cfe_es_events.h, 1378
CFE_ES_SYSLOGMODE_EID
cfe_es_events.h, 1378
CFE_ES_SendMemPoolStatsCmd, 710
CommandHeader, 710
Payload, 710
CFE_ES_SendMemPoolStatsCmd_Payload, 711
Application, 711
PoolHandle, 711
CFE_ES_SendMemPoolStatsCmd_Payload_t
cfe_es_msg.h, 1414
CFE_ES_SendMemPoolStatsCmd_t
cfe_es_msg.h, 1415
CFE_ES_SetGenCount
cFE Generic Counter APIs, 394
CFE_ES_SetMaxPRCountCmd, 712
CommandHeader, 712
Payload, 712
CFE_ES_SetMaxPRCountCmd_Payload, 713
MaxPRCount, 713
CFE_ES_SetMaxPRCountCmd_Payload_t
cfe_es_msg.h, 1415
CFE_ES_SetMaxPRCountCmd_t
cfe_es_msg.h, 1415
CFE_ES_SetPerfFilterMaskCmd, 713
CommandHeader, 714
Payload, 714
CFE_ES_SetPerfFilterMaskCmd_Payload, 714
FilterMask, 715
FilterMaskNum, 715
CFE_ES_SetPerfFilterMaskCmd_Payload_t
cfe_es_msg.h, 1415
CFE_ES_SetPerfFilterMaskCmd_t
cfe_es_msg.h, 1415
CFE_ES_SetPerfTrigMaskCmd_Payload, 716
TriggerMask, 717
TriggerMaskNum, 717
CFE_ES_SetPerfTrigMaskCmd_Payload_t
cfe_es_msg.h, 1416
CFE_ES_SetPerfTriggerMaskCmd, 715
CommandHeader, 716
Payload, 716
CFE_ES_SetPerfTriggerMaskCmd_t
cfe_es_msg.h, 1415
CFE_ES_StackPointer_t
cfe_es_api_typedefs.h, 1268
CFE_ES_StartApp, 717
CommandHeader, 718
Payload, 718
CFE_ES_StartAppCmd_Payload, 718
AppEntryPoint, 719
AppFileName, 719
Application, 719
ExceptionAction, 720
Priority, 720
StackSize, 720
CFE_ES_StartAppCmd_Payload_t
cfe_es_msg.h, 1416
CFE_ES_StartAppCmd_t
cfe_es_msg.h, 1416
CFE_ES_StartPerfCmd_Payload, 720
TriggerMode, 721
CFE_ES_StartPerfCmd_Payload_t
cfe_es_msg.h, 1416
CFE_ES_StartPerfDataCmd, 721
CommandHeader, 722
Payload, 722
CFE_ES_StartPerfDataCmd_t
cfe_es_msg.h, 1416

CFE_ES_StopAppCmd_t
cfe_es_msg.h, 1416
CFE_ES_StopPerfCmd_Payload, 722
DataFileName, 723
CFE_ES_StopPerfCmd_Payload_t
cfe_es_msg.h, 1417
CFE_ES_StopPerfDataCmd, 723
CommandHeader, 724
Payload, 724
CFE_ES_StopPerfDataCmd_t
cfe_es_msg.h, 1417
CFE_ES_SystemState
cfe_es_extern_typedefs.h, 1280
CFE_ES_SystemState_Enum_t
cfe_es_extern_typedefs.h, 1276
CFE_ES_TASK_DELETE_ERR
cFE Return Code Defines, 316
CFE_ES_TASK_STACK_ALLOCATE
cfe_es_api_typedefs.h, 1266
CFE_ES_TASKID_UNDEFINED
cfe_es_api_typedefs.h, 1267
CFE_ES_TASKID_C
cfe_es_api_typedefs.h, 1267
CFE_ES_TASKINFO_EID
cfe_es_events.h, 1378
CFE_ES_TASKINFO_OSCREATE_ERR_EID
cfe_es_events.h, 1379
CFE_ES_TASKINFO_WR_ERR_EID
cfe_es_events.h, 1379
CFE_ES_TASKINFO_WRHDR_ERR_EID
cfe_es_events.h, 1379
CFE_ES_TASKWR_ERR_EID
cfe_es_events.h, 1380
CFE_ES_TEST_LONG_MASK
cfe_es.h, 1262
CFE_ES_TIMER_DELETE_ERR
cFE Return Code Defines, 316
CFE_ES_TLM_POOL_STATS_INFO_EID
cfe_es_events.h, 1380
CFE_ES_TaskEntryFuncPtr_t
cfe_es_api_typedefs.h, 1268
CFE_ES_TaskID_ToIndex
cFE Resource ID APIs, 341
CFE_ES_TaskId_t
cfe_es_extern_typedefs.h, 1277
CFE_ES_TaskInfo, 724
AppId, 725
AppName, 725
ExecutionCounter, 725
Priority, 725
Spare, 726
StackSize, 726
TaskId, 726
TaskName, 726
CFE_ES_TaskInfo_t
cfe_es_extern_typedefs.h, 1277
CFE_ES_TaskPriority_Atom_t
cfe_es_extern_typedefs.h, 1277
CFE_ES_USE_MUTEX
cfe_es_api_typedefs.h, 1267
CFE_ES_VERSION_INF_EID
cfe_es_events.h, 1380
CFE_ES_WRHDR_ERR_EID
cfe_es_events.h, 1381
CFE_ES_WRITE_CFE_HDR_ERR_EID
cfe_es_events.h, 1381
CFE_ES_WRITE_ER_LOG_CC
cfe_es_msg.h, 1407
CFE_ES_WRITE_SYSLOG_CC
cfe_es_msg.h, 1408
CFE_ES_WaitForStartupSync
cFE Application Behavior APIs, 351
CFE_ES_WaitForSystemState
cFE Application Behavior APIs, 351
CFE_ES_WriteERLogCmd_t
cfe_es_msg.h, 1417
CFE_ES_WriteSysLogCmd_t
cfe_es_msg.h, 1417
CFE_ES_WriteToSysLog
cFE Miscellaneous APIs, 370
CFE_EVENTS_SERVICE
cfe_error.h, 1255
CFE_EVS_ADD_EVENT_FILTER_CC
cfe_evs_msg.h, 1436
CFE_EVS_ADDFILTER_EID
cfe_evs_events.h, 1419
CFE_EVS_APP_FILTER_OVERLOAD
cFE Return Code Defines, 317
CFE_EVS_APP_ILLEGAL_APP_ID
cFE Return Code Defines, 317
CFE_EVS_APP_NOT_REGISTERED
cFE Return Code Defines, 317
CFE_EVS_AddEventFilterCmd_t
cfe_evs_msg.h, 1457
CFE_EVS_AppDataCmd_Payload, 727
AppDataFilename, 727
CFE_EVS_AppDataCmd_Payload_t
cfe_evs_msg.h, 1457
CFE_EVS_AppNameBitMaskCmd, 727
CommandHeader, 728
Payload, 728
CFE_EVS_AppNameBitMaskCmd_Payload, 728
AppName, 729
BitMask, 729
Spare, 729
CFE_EVS_AppNameBitMaskCmd_Payload_t
cfe_evs_msg.h, 1458
CFE_EVS_AppNameBitMaskCmd_t

cfe_evs_msg.h, 1458
CFE_EVS_AppNameCmd, 730
CommandHeader, 730
Payload, 730
CFE_EVS_AppNameCmd_Payload, 731
AppName, 731
CFE_EVS_AppNameCmd_Payload_t
cfe_evs_msg.h, 1458
CFE_EVS_AppNameCmd_t
cfe_evs_msg.h, 1458
CFE_EVS_AppNameEventIDCmd, 732
CommandHeader, 732
Payload, 732
CFE_EVS_AppNameEventIDCmd_Payload, 733
AppName, 733
EventID, 733
CFE_EVS_AppNameEventIDCmd_Payload_t
cfe_evs_msg.h, 1458
CFE_EVS_AppNameEventIDCmd_t
cfe_evs_msg.h, 1458
CFE_EVS_AppNameEventIDMaskCmd, 734
CommandHeader, 734
Payload, 734
CFE_EVS_AppNameEventIDMaskCmd_Payload, 735
AppName, 735
EventID, 735
Mask, 735
CFE_EVS_AppNameEventIDMaskCmd_Payload_t
cfe_evs_msg.h, 1459
CFE_EVS_AppNameEventIDMaskCmd_t
cfe_evs_msg.h, 1459
CFE_EVS_AppTlmData, 736
AppEnableStatus, 736
AppID, 736
AppMessageSentCounter, 737
Padding, 737
CFE_EVS_AppTlmData_t
cfe_evs_msg.h, 1459
CFE_EVS_BinFilter, 737
EventID, 738
Mask, 738
CFE_EVS_BinFilter_t
cfe_evs_api_typedefs.h, 1286
CFE_EVS_BitMaskCmd, 738
CommandHeader, 739
Payload, 739
CFE_EVS_BitMaskCmd_Payload, 739
BitMask, 740
Spare, 740
CFE_EVS_BitMaskCmd_Payload_t
cfe_evs_msg.h, 1459
CFE_EVS_BitMaskCmd_t
cfe_evs_msg.h, 1459
CFE_EVS_CLEAR_LOG_CC
cfe_evs_msg.h, 1436
CFE_EVS_CMD_MID
cpu1_msgids.h, 1155
CFE_EVS_CRITICAL_BIT
cfe_evs_msg.h, 1437
CFE_EVS_ClearLogCmd_t
cfe_evs_msg.h, 1459
CFE_EVS_DEBUG_BIT
cfe_evs_msg.h, 1437
CFE_EVS_DELETE_EVENT_FILTER_CC
cfe_evs_msg.h, 1438
CFE_EVS_DELFILTER_EID
cfe_evs_events.h, 1419
CFE_EVS_DISABLE_APP_EVENT_TYPE_CC
cfe_evs_msg.h, 1438
CFE_EVS_DISABLE_APP_EVENTS_CC
cfe_evs_msg.h, 1439
CFE_EVS_DISABLE_EVENT_TYPE_CC
cfe_evs_msg.h, 1440
CFE_EVS_DISABLE_PORTS_CC
cfe_evs_msg.h, 1441
CFE_EVS_DISAPPENTTYPE_EID
cfe_evs_events.h, 1420
CFE_EVS_DISAPPEVT_EID
cfe_evs_events.h, 1420
CFE_EVS_DISEVTTYPE_EID
cfe_evs_events.h, 1420
CFE_EVS_DISPORT_EID
cfe_evs_events.h, 1421
CFE_EVS_DeleteEventFilterCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_DisableAppEventTypeCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_DisableAppEventsCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_DisableEventTypeCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_DisablePortsCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_ENAAPPEVT_EID
cfe_evs_events.h, 1421
CFE_EVS_ENAAPPEVTTYPE_EID
cfe_evs_events.h, 1421
CFE_EVS_ENABLE_APP_EVENT_TYPE_CC
cfe_evs_msg.h, 1442
CFE_EVS_ENABLE_APP_EVENTS_CC
cfe_evs_msg.h, 1443
CFE_EVS_ENABLE_EVENT_TYPE_CC
cfe_evs_msg.h, 1444
CFE_EVS_ENABLE_PORTS_CC
cfe_evs_msg.h, 1445
CFE_EVS_ENAEVTTYPE_EID
cfe_evs_events.h, 1422
CFE_EVS_ENAPORT_EID

cfe_evs_events.h, 1422
CFE_EVS_ERR_APPNOREGS_EID
cfe_evs_events.h, 1422
CFE_EVS_ERR_CC_EID
cfe_evs_events.h, 1423
CFE_EVS_ERR_CRDATFILE_EID
cfe_evs_events.h, 1423
CFE_EVS_ERR_CRLOGFILE_EID
cfe_evs_events.h, 1423
CFE_EVS_ERR_EVTIDNOREGS_EID
cfe_evs_events.h, 1424
CFE_EVS_ERR_ILLAPPIDRANGE_EID
cfe_evs_events.h, 1424
CFE_EVS_ERR_ILLEGALFMTMOD_EID
cfe_evs_events.h, 1424
CFE_EVS_ERR_INVALID_BITMASK_EID
cfe_evs_events.h, 1425
CFE_EVS_ERR_LOGMODE_EID
cfe_evs_events.h, 1425
CFE_EVS_ERR_MAXREGSFILTER_EID
cfe_evs_events.h, 1425
CFE_EVS_ERR_MSGID_EID
cfe_evs_events.h, 1426
CFE_EVS_ERR_NOAPPIDFOUND_EID
cfe_evs_events.h, 1426
CFE_EVS_ERR_UNREGISTERED_EVS_APP
cfe_evs_events.h, 1426
CFE_EVS_ERR_WRDATFILE_EID
cfe_evs_events.h, 1427
CFE_EVS_ERR_WRLOGFILE_EID
cfe_evs_events.h, 1427
CFE_EVS_ERROR_BIT
cfe_evs_msg.h, 1446
CFE_EVS_EVERY_FOURTH_ONE
cfe_evs_api_typedefs.h, 1284
CFE_EVS_EVERY_OTHER_ONE
cfe_evs_api_typedefs.h, 1284
CFE_EVS_EVERY_OTHER_TWO
cfe_evs_api_typedefs.h, 1284
CFE_EVS_EVT_FILTERED_EID
cfe_evs_events.h, 1427
CFE_EVS_EVT_NOT_REGISTERED
cFE Return Code Defines, 317
CFE_EVS_EnableAppEventTypeCmd_t
cfe_evs_msg.h, 1461
CFE_EVS_EnableAppEventsCmd_t
cfe_evs_msg.h, 1460
CFE_EVS_EnableEventTypeCmd_t
cfe_evs_msg.h, 1461
CFE_EVS_EnablePortsCmd_t
cfe_evs_msg.h, 1461
CFE_EVS_EventFilter
cfe_evs_extern_typedefs.h, 1288
CFE_EVS_EventFilter_Enum_t
cfe_evs_extern_typedefs.h, 1287
CFE_EVS_EventOutput
cfe_evs_extern_typedefs.h, 1288
CFE_EVS_EventOutput_Enum_t
cfe_evs_extern_typedefs.h, 1287
CFE_EVS_EventType
cfe_evs_extern_typedefs.h, 1290
CFE_EVS_EventType_Enum_t
cfe_evs_extern_typedefs.h, 1287
CFE_EVS_FILE_WRITE_ERROR
cFE Return Code Defines, 318
CFE_EVS_FILTER_MAX_EID
cfe_evs_events.h, 1428
CFE_EVS_FIRST_16_STOP
cfe_evs_api_typedefs.h, 1284
CFE_EVS_FIRST_32_STOP
cfe_evs_api_typedefs.h, 1284
CFE_EVS_FIRST_4_STOP
cfe_evs_api_typedefs.h, 1284
CFE_EVS_FIRST_64_STOP
cfe_evs_api_typedefs.h, 1285
CFE_EVS_FIRST_8_STOP
cfe_evs_api_typedefs.h, 1285
CFE_EVS_FIRST_ONE_STOP
cfe_evs_api_typedefs.h, 1285
CFE_EVS_FIRST_TWO_STOP
cfe_evs_api_typedefs.h, 1285
CFE_EVS_HK_TLM_MID
cpu1_msgids.h, 1155
CFE_EVS_HousekeepingTlm, 740
Payload, 741
TelemetryHeader, 741
CFE_EVS_HousekeepingTlm_Payload, 741
AppData, 742
CommandCounter, 743
CommandErrorCounter, 743
LogEnabled, 743
LogFullFlag, 743
LogMode, 744
LogOverflowCounter, 744
MessageFormatMode, 744
MessageSendCounter, 744
MessageTruncCounter, 745
OutputPort, 745
Spare1, 745
Spare2, 745
Spare3, 746
UnregisteredAppCounter, 746
CFE_EVS_HousekeepingTlm_Payload_t
cfe_evs_msg.h, 1461
CFE_EVS_HousekeepingTlm_t
cfe_evs_msg.h, 1461
CFE_EVS_INFORMATION_BIT
cfe_evs_msg.h, 1447

CFE_EVS_INVALID_PARAMETER
cFE Return Code Defines, 318
CFE_EVS_LEN_ERR_EID
cfe_evs_events.h, 1428
CFE_EVS_LOGMODE_EID
cfe_evs_events.h, 1428
CFE_EVS_LONG_EVENT_MSG_MID
cpu1_msgids.h, 1156
CFE_EVS_LogFileCmd_Payload, 746
LogFilename, 747
CFE_EVS_LogFileCmd_Payload_t
cfe_evs_msg.h, 1461
CFE_EVS_LogMode
cfe_evs_extern_typedefs.h, 1290
CFE_EVS_LogMode_Enum_t
cfe_evs_extern_typedefs.h, 1287
CFE_EVS_LongEventTlm, 747
Payload, 748
TelemetryHeader, 748
CFE_EVS_LongEventTlm_Payload, 748
Message, 749
PacketID, 749
Spare1, 749
Spare2, 749
CFE_EVS_LongEventTlm_Payload_t
cfe_evs_msg.h, 1462
CFE_EVS_LongEventTlm_t
cfe_evs_msg.h, 1462
CFE_EVS_MsgFormat
cfe_evs_extern_typedefs.h, 1290
CFE_EVS_MsgFormat_Enum_t
cfe_evs_extern_typedefs.h, 1288
CFE_EVS_NO_FILTER
cfe_evs_api_typedefs.h, 1285
CFE_EVS_NOOP_CC
cfe_evs_msg.h, 1447
CFE_EVS_NOOP_EID
cfe_evs_events.h, 1429
CFE_EVS_NOT_IMPLEMENTED
cFE Return Code Defines, 318
CFE_EVS_NoArgsCmd, 750
CommandHeader, 750
CFE_EVS_NoArgsCmd_t
cfe_evs_msg.h, 1462
CFE_EVS_NoopCmd_t
cfe_evs_msg.h, 1462
CFE_EVS_PORT1_BIT
cfe_evs_msg.h, 1447
CFE_EVS_PORT2_BIT
cfe_evs_msg.h, 1448
CFE_EVS_PORT3_BIT
cfe_evs_msg.h, 1448
CFE_EVS_PORT4_BIT
cfe_evs_msg.h, 1448
CFE_EVS_PacketID_t
cfe_evs_msg.h, 1462
CFE_EVS_PacketID, 750
AppName, 751
EventID, 751
EventType, 751
ProcessorID, 751
SpacecraftID, 752
CFE_EVS_RESET_ALL_FILTERS_CC
cfe_evs_msg.h, 1448
CFE_EVS_RESET_APP_COUNTER_CC
cfe_evs_msg.h, 1449
CFE_EVS_RESET_AREA_POINTER
cFE Return Code Defines, 318
CFE_EVS_RESET_COUNTERS_CC
cfe_evs_msg.h, 1450
CFE_EVS_RESET_FILTER_CC
cfe_evs_msg.h, 1451
CFE_EVS_RSTALLFILTER_EID
cfe_evs_events.h, 1429
CFE_EVS_RSTCNT_EID
cfe_evs_events.h, 1429
CFE_EVS_RSTEVTCNT_EID
cfe_evs_events.h, 1430
CFE_EVS_RSTFILTER_EID
cfe_evs_events.h, 1430
CFE_EVS_Register
cFE Registration APIs, 396
CFE_EVS_ResetAllFilters
cFE Reset Event Filter APIs, 403
CFE_EVS_ResetAllFiltersCmd_t
cfe_evs_msg.h, 1462
CFE_EVS_ResetAppCounterCmd_t
cfe_evs_msg.h, 1462
CFE_EVS_ResetCountersCmd_t
cfe_evs_msg.h, 1463
CFE_EVS_ResetFilter
cFE Reset Event Filter APIs, 403
CFE_EVS_ResetFilterCmd_t
cfe_evs_msg.h, 1463
CFE_EVS_SEND_HK_MID
cpu1_msgids.h, 1156
CFE_EVS_SET_EVENT_FORMAT_MODE_CC
cfe_evs_msg.h, 1452
CFE_EVS_SET_FILTER_CC
cfe_evs_msg.h, 1453
CFE_EVS_SET_LOG_MODE_CC
cfe_evs_msg.h, 1454
CFE_EVS_SETEVTFMTMOD_EID
cfe_evs_events.h, 1430
CFE_EVS_SETFILTERMSK_EID
cfe_evs_events.h, 1431
CFE_EVS_SHORT_EVENT_MSG_MID
cpu1_msgids.h, 1156

CFE_EVS_STARTUP_EID
cfe_evs_events.h, 1431
CFE_EVS_Send
cfe_evs.h, 1281
CFE_EVS_SendCrit
cfe_evs.h, 1282
CFE_EVS_SendDbg
cfe_evs.h, 1282
CFE_EVS_SendErr
cfe_evs.h, 1282
CFE_EVS_SendEvent
cFE Send Event APIs, 398
CFE_EVS_SendEventWithAppID
cFE Send Event APIs, 399
CFE_EVS_SendInfo
cfe_evs.h, 1282
CFE_EVS_SendTimedEvent
cFE Send Event APIs, 401
CFE_EVS_SetEventFormatCode_Payload, 752
MsgFormat, 753
Spare, 753
CFE_EVS_SetEventFormatMode_Payload_t
cfe_evs_msg.h, 1463
CFE_EVS_SetEventFormatModeCmd, 753
CommandHeader, 754
Payload, 754
CFE_EVS_SetEventFormatModeCmd_t
cfe_evs_msg.h, 1463
CFE_EVS_SetFilterCmd_t
cfe_evs_msg.h, 1463
CFE_EVS_SetLogMode_Payload, 754
LogMode, 755
Spare, 755
CFE_EVS_SetLogMode_Payload_t
cfe_evs_msg.h, 1463
CFE_EVS_SetLogModeCmd, 755
CommandHeader, 756
Payload, 756
CFE_EVS_SetLogModeCmd_t
cfe_evs_msg.h, 1464
CFE_EVS_ShortEventTlm, 756
Payload, 756
TelemetryHeader, 757
CFE_EVS_ShortEventTlm_Payload, 757
PacketID, 757
CFE_EVS_ShortEventTlm_Payload_t
cfe_evs_msg.h, 1464
CFE_EVS_ShortEventTlm_t
cfe_evs_msg.h, 1464
CFE_EVS_UNKNOWN_FILTER
cFE Return Code Defines, 319
CFE_EVS_WRDAT_EID
cfe_evs_events.h, 1431
CFE_EVS_WRITE_APP_DATA_FILE_CC
cfe_evs_msg.h, 1455
CFE_EVS_WRITE_LOG_DATA_FILE_CC
cfe_evs_msg.h, 1456
CFE_EVS_WRLOG_EID
cfe_evs_events.h, 1432
CFE_EVS_WriteAppDataFileCmd, 758
CommandHeader, 758
Payload, 758
CFE_EVS_WriteAppDataFileCmd_t
cfe_evs_msg.h, 1464
CFE_EVS_WriteLogDataFileCmd, 759
CommandHeader, 759
Payload, 759
CFE_EVS_WriteLogDataFileCmd_t
cfe_evs_msg.h, 1464
CFE_EXECUTIVE_SERVICE
cfe_error.h, 1255
CFE_FILE_SERVICE
cfe_error.h, 1256
CFE_FS_BAD_ARGUMENT
cFE Return Code Defines, 319
CFE_FS_BackgroundFileDumpIsPending
cFE File Utility APIs, 410
CFE_FS_BackgroundFileDumpRequest
cFE File Utility APIs, 411
CFE_FS_ExtractFilenameFromPath
cFE File Utility APIs, 411
CFE_FS_FILE_CONTENT_ID
cfe_fs_extern_typedefs.h, 1296
CFE_FS_FNAME_TOO_LONG
cFE Return Code Defines, 319
CFE_FS_FileCategory_t
cfe_fs_api_typedefs.h, 1294
CFE_FS_FileWriteEvent_t
cfe_fs_api_typedefs.h, 1295
CFE_FS_FileWriteGetData_t
cfe_fs_api_typedefs.h, 1293
CFE_FS_FileWriteMetaData, 760
Description, 760
FileName, 760
FileSubType, 761
GetData, 761
IsPending, 761
OnEvent, 761
CFE_FS_FileWriteMetaData_t
cfe_fs_api_typedefs.h, 1293
CFE_FS_FileWriteOnEvent_t
cfe_fs_api_typedefs.h, 1293
CFE_FS_GetDefaultExtension
cFE File Utility APIs, 412
CFE_FS_GetDefaultMountPoint
cFE File Utility APIs, 413
CFE_FS_HDR_DESC_MAX_LEN
cfe_fs_extern_typedefs.h, 1296

CFE_FS_Header, 762
ApplicationID, 762
ContentType, 762
Description, 763
Length, 763
ProcessorID, 763
SpacecraftID, 763
SubType, 763
TimeSeconds, 764
TimeSubSeconds, 764
CFE_FS_Header_t
cfe_fs_extern_typedefs.h, 1296
CFE_FS_INVALID_PATH
cFE Return Code Defines, 319
CFE_FS_InitHeader
cFE File Header Management APIs, 405
CFE_FS_NOT_IMPLEMENTED
cFE Return Code Defines, 320
CFE_FS_ParseInputFileName
cFE File Utility APIs, 413
CFE_FS_ParseInputFileNameEx
cFE File Utility APIs, 414
CFE_FS_ReadHeader
cFE File Header Management APIs, 405
CFE_FS_SetTimestamp
cFE File Header Management APIs, 406
CFE_FS_SubType
cfe_fs_extern_typedefs.h, 1297
CFE_FS_SubType_Enum_t
cfe_fs_extern_typedefs.h, 1296
CFE_FS_WriteHeader
cFE File Header Management APIs, 407
CFE_GENERIC_SERVICE
cfe_error.h, 1256
CFE_MAJOR_VERSION
cfe_version.h, 1347
CFE_MAKE_BIG16
cfe_endian.h, 1249
CFE_MAKE_BIG32
cfe_endian.h, 1249
CFE_MINOR_VERSION
cfe_version.h, 1347
CFE_MISSION_ES_APP_TLM_MSG
sample_mission_cfg.h, 1220
CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN
sample_mission_cfg.h, 1220
CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
sample_mission_cfg.h, 1221
CFE_MISSION_ES_CMD_MSG
sample_mission_cfg.h, 1221
CFE_MISSION_ES_CRC_16
sample_mission_cfg.h, 1222
CFE_MISSION_ES_CRC_32
sample_mission_cfg.h, 1222
CFE_MISSION_ES_CRC_8
sample_mission_cfg.h, 1222
CFE_MISSION_ES_DEFAULT_CRC
sample_mission_cfg.h, 1222
CFE_MISSION_ES_HK_TLM_MSG
sample_mission_cfg.h, 1223
CFE_MISSION_ES_MAIN_PERF_ID
sample_perfids.h, 1239
CFE_MISSION_ES_MAX_APPLICATIONS
sample_mission_cfg.h, 1223
CFE_MISSION_ES_MEMSTATS_TLM_MSG
sample_mission_cfg.h, 1223
CFE_MISSION_ES_PERF_EXIT_BIT
sample_perfids.h, 1240
CFE_MISSION_ES_PERF_MAX_IDS
sample_mission_cfg.h, 1224
CFE_MISSION_ES_POOL_MAX_BUCKETS
sample_mission_cfg.h, 1224
CFE_MISSION_ES_SEND_HK_MSG
sample_mission_cfg.h, 1224
CFE_MISSION_EVS_CMD_MSG
sample_mission_cfg.h, 1225
CFE_MISSION_EVS_HK_TLM_MSG
sample_mission_cfg.h, 1225
CFE_MISSION_EVS_LONG_EVENT_MSG_MSG
sample_mission_cfg.h, 1225
CFE_MISSION_EVS_MAIN_PERF_ID
sample_perfids.h, 1240
CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
sample_mission_cfg.h, 1225
CFE_MISSION_EVS_SEND_HK_MSG
sample_mission_cfg.h, 1226
CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG
sample_mission_cfg.h, 1226
CFE_MISSION_MAX_API_LEN
sample_mission_cfg.h, 1226
CFE_MISSION_MAX_FILE_LEN
sample_mission_cfg.h, 1226
CFE_MISSION_MAX_PATH_LEN
sample_mission_cfg.h, 1227
CFE_MISSION_REV
cfe_version.h, 1347
CFE_MISSION_SB_ALLSUBS_TLM_MSG
sample_mission_cfg.h, 1227
CFE_MISSION_SB_CMD_MSG
sample_mission_cfg.h, 1228
CFE_MISSION_SB_HK_TLM_MSG
sample_mission_cfg.h, 1228
CFE_MISSION_SB_MAIN_PERF_ID
sample_perfids.h, 1240
CFE_MISSION_SB_MAX_PIPES
sample_mission_cfg.h, 1228
CFE_MISSION_SB_MAX_SB_MSG_SIZE
sample_mission_cfg.h, 1228

CFE_MISSION_SB_MSG_LIM_PERF_ID
sample_perfids.h, 1240
CFE_MISSION_SB_ONESUB_TLM_MSG
sample_mission_cfg.h, 1229
CFE_MISSION_SB_PIPE_OFLOW_PERF_ID
sample_perfids.h, 1240
CFE_MISSION_SB_SEND_HK_MSG
sample_mission_cfg.h, 1229
CFE_MISSION_SB_STATS_TLM_MSG
sample_mission_cfg.h, 1229
CFE_MISSION_SB_SUB_RPT_CTRL_MSG
sample_mission_cfg.h, 1229
CFE_MISSION_TBL_CMD_MSG
sample_mission_cfg.h, 1230
CFE_MISSION_TBL_HK_TLM_MSG
sample_mission_cfg.h, 1230
CFE_MISSION_TBL_MAIN_PERF_ID
sample_perfids.h, 1241
CFE_MISSION_TBL_MAX_FULL_NAME_LEN
sample_mission_cfg.h, 1230
CFE_MISSION_TBL_MAX_NAME_LENGTH
sample_mission_cfg.h, 1230
CFE_MISSION_TBL_REG_TLM_MSG
sample_mission_cfg.h, 1231
CFE_MISSION_TBL_SEND_HK_MSG
sample_mission_cfg.h, 1231
CFE_MISSION_TEST_CMD_MSG
sample_mission_cfg.h, 1231
CFE_MISSION_TEST_HK_TLM_MSG
sample_mission_cfg.h, 1231
CFE_MISSION_TIME_1HZ_CMD_MSG
sample_mission_cfg.h, 1232
CFE_MISSION_TIME_AT_TONE_WAS
sample_mission_cfg.h, 1232
CFE_MISSION_TIME_AT_TONE_WILL_BE
sample_mission_cfg.h, 1232
CFE_MISSION_TIME_CFG_DEFAULT_TAI
sample_mission_cfg.h, 1232
CFE_MISSION_TIME_CFG_DEFAULT_UTC
sample_mission_cfg.h, 1233
CFE_MISSION_TIME_CFG_FAKE_TONE
sample_mission_cfg.h, 1233
CFE_MISSION_TIME_CMD_MSG
sample_mission_cfg.h, 1233
CFE_MISSION_TIME_DATA_CMD_MSG
sample_mission_cfg.h, 1234
CFE_MISSION_TIME_DEF_DELAY_SECS
sample_mission_cfg.h, 1234
CFE_MISSION_TIME_DEF_DELAY_SUBS
sample_mission_cfg.h, 1234
CFE_MISSION_TIME_DEF_LEAPS
sample_mission_cfg.h, 1234
CFE_MISSION_TIME_DEF_MET_SECS
sample_mission_cfg.h, 1234
CFE_MISSION_TIME_DEF_MET_SUBS
sample_mission_cfg.h, 1235
CFE_MISSION_TIME_DEF_STCF_SECS
sample_mission_cfg.h, 1235
CFE_MISSION_TIME_DEF_STCF_SUBS
sample_mission_cfg.h, 1235
CFE_MISSION_TIME_DIAG_TLM_MSG
sample_mission_cfg.h, 1235
CFE_MISSION_TIME_EPOCH_DAY
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_EPOCH_HOUR
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_EPOCH_MINUTE
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_EPOCH_SECOND
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_EPOCH_YEAR
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_FS_FACTOR
sample_mission_cfg.h, 1236
CFE_MISSION_TIME_HK_TLM_MSG
sample_mission_cfg.h, 1237
CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID
sample_perfids.h, 1241
CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID
sample_perfids.h, 1241
CFE_MISSION_TIME_MAIN_PERF_ID
sample_perfids.h, 1241
CFE_MISSION_TIME_MAX_ELAPSED
sample_mission_cfg.h, 1237
CFE_MISSION_TIME_MIN_ELAPSED
sample_mission_cfg.h, 1237
CFE_MISSION_TIME_SEND_CMD_MSG
sample_mission_cfg.h, 1238
CFE_MISSION_TIME_SEND_HK_MSG
sample_mission_cfg.h, 1238
CFE_MISSION_TIME_SENDMET_PERF_ID
sample_perfids.h, 1241
CFE_MISSION_TIME_TONE1HZISR_PERF_ID
sample_perfids.h, 1242
CFE_MISSION_TIME_TONE1HZTASK_PERF_ID
sample_perfids.h, 1242
CFE_MISSION_TIME_TONE_CMD_MSG
sample_mission_cfg.h, 1238
CFE_MSG_ApId_t
cfe_msg_api_typedefs.h, 1302
CFE_MSG_BAD_ARGUMENT
cfe_msg_api_typedefs.h, 1302
CFE_MSG_Checksum_t
cfe_msg_api_typedefs.h, 1303
CFE_MSG_CommandHeader_t
cfe_msg_api_typedefs.h, 1303
CFE_MSG_EDSVersion_t
cfe_msg_api_typedefs.h, 1303

CFE_MSG_Endian
cfe_msg_api_typedefs.h, 1305
CFE_MSG_Endian_t
cfe_msg_api_typedefs.h, 1303
CFE_MSG_FcnCode_t
cfe_msg_api_typedefs.h, 1303
CFE_MSG_GenerateChecksum
cFE Message Secondary Header APIs, 435
CFE_MSG_GetApId
cFE Message Primary Header APIs, 417
CFE_MSG_GetEDSVersion
cFE Message Extended Header APIs, 428
CFE_MSG_GetEndian
cFE Message Extended Header APIs, 429
CFE_MSG_GetFcnCode
cFE Message Secondary Header APIs, 436
CFE_MSG_GetHasSecondaryHeader
cFE Message Primary Header APIs, 418
CFE_MSG_GetHeaderVersion
cFE Message Primary Header APIs, 419
CFE_MSG_GetMsgId
cFE Message Id APIs, 441
CFE_MSG_GetMsgTime
cFE Message Secondary Header APIs, 436
CFE_MSG_GetNextSequenceCount
cFE Message Primary Header APIs, 419
CFE_MSG_GetPlaybackFlag
cFE Message Extended Header APIs, 429
CFE_MSG_GetSegmentationFlag
cFE Message Primary Header APIs, 420
CFE_MSG_GetSequenceCount
cFE Message Primary Header APIs, 420
CFE_MSG_GetSize
cFE Message Primary Header APIs, 421
CFE_MSG_GetSubsystem
cFE Message Extended Header APIs, 430
CFE_MSG_GetSystem
cFE Message Extended Header APIs, 431
CFE_MSG_GetType
cFE Message Primary Header APIs, 422
CFE_MSG_GetTypeFromMsgId
cFE Message Id APIs, 442
CFE_MSG_HeaderVersion_t
cfe_msg_api_typedefs.h, 1304
CFE_MSG_Init
cFE Generic Message APIs, 416
CFE_MSG_Message_t
cfe_msg_api_typedefs.h, 1304
CFE_MSG_NOT_IMPLEMENTED
cfe_msg_api_typedefs.h, 1302
CFE_MSG_PlaybackFlag
cfe_msg_api_typedefs.h, 1306
CFE_MSG_PlaybackFlag_t
cfe_msg_api_typedefs.h, 1304
CFE_MSG_SegmentationFlag
cfe_msg_api_typedefs.h, 1306
CFE_MSG_SegmentationFlag_t
cfe_msg_api_typedefs.h, 1304
CFE_MSG_SequenceCount_t
cfe_msg_api_typedefs.h, 1304
CFE_MSG_SetApId
cFE Message Primary Header APIs, 422
CFE_MSG_SetEDSVersion
cFE Message Extended Header APIs, 431
CFE_MSG_SetEndian
cFE Message Extended Header APIs, 432
CFE_MSG_SetFcnCode
cFE Message Secondary Header APIs, 437
CFE_MSG_SetHasSecondaryHeader
cFE Message Primary Header APIs, 424
CFE_MSG_SetHeaderVersion
cFE Message Primary Header APIs, 424
CFE_MSG_SetMsgId
cFE Message Id APIs, 442
CFE_MSG_SetMsgTime
cFE Message Secondary Header APIs, 439
CFE_MSG_SetPlaybackFlag
cFE Message Extended Header APIs, 433
CFE_MSG_SetSegmentationFlag
cFE Message Primary Header APIs, 425
CFE_MSG_SetSequenceCount
cFE Message Primary Header APIs, 426
CFE_MSG_SetSize
cFE Message Primary Header APIs, 426
CFE_MSG_SetSubsystem
cFE Message Extended Header APIs, 433
CFE_MSG_SetSystem
cFE Message Extended Header APIs, 434
CFE_MSG_SetType
cFE Message Primary Header APIs, 427
CFE_MSG_Size_t
cfe_msg_api_typedefs.h, 1305
CFE_MSG_Subsystem_t
cfe_msg_api_typedefs.h, 1305
CFE_MSG_System_t
cfe_msg_api_typedefs.h, 1305
CFE_MSG_TelemetryHeader_t
cfe_msg_api_typedefs.h, 1305
CFE_MSG_Type
cfe_msg_api_typedefs.h, 1306
CFE_MSG_Type_t
cfe_msg_api_typedefs.h, 1305
CFE_MSG_ValidateChecksum
cFE Message Secondary Header APIs, 440
CFE_MSG_WRONG_MSG_TYPE
cfe_msg_api_typedefs.h, 1302
CFE_PLATFORM_CMD_MID_BASE_GLOB
cpu1_msgids.h, 1156

CFE_PLATFORM_CMD_MID_BASE
cpu1_msgids.h, 1156
CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
cpu1_platform_cfg.h, 1164
CFE_PLATFORM_ENDIAN
cpu1_platform_cfg.h, 1165
CFE_PLATFORM_ES_APP_KILL_TIMEOUT
cpu1_platform_cfg.h, 1165
CFE_PLATFORM_ES_APP_SCAN_RATE
cpu1_platform_cfg.h, 1166
CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE
cpu1_platform_cfg.h, 1166
CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
cpu1_platform_cfg.h, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
cpu1_platform_cfg.h, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02
cpu1_platform_cfg.h, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08
cpu1_platform_cfg.h, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14
cpu1_platform_cfg.h, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15
cpu1_platform_cfg.h, 1170
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16
cpu1_platform_cfg.h, 1170
CFE_PLATFORM_ES_CDS_SIZE
cpu1_platform_cfg.h, 1170
CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
cpu1_platform_cfg.h, 1170
CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_↩
FILE
cpu1_platform_cfg.h, 1171
CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
cpu1_platform_cfg.h, 1171
CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILE↩
NAME
cpu1_platform_cfg.h, 1171
CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG_MO↩
DE
cpu1_platform_cfg.h, 1172
CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE
cpu1_platform_cfg.h, 1172
CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
cpu1_platform_cfg.h, 1173
CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE
cpu1_platform_cfg.h, 1173
CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE
cpu1_platform_cfg.h, 1174
CFE_PLATFORM_ES_ER_LOG_ENTRIES
cpu1_platform_cfg.h, 1174
CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE
cpu1_platform_cfg.h, 1175
CFE_PLATFORM_ES_MAX_APPLICATIONS
cpu1_platform_cfg.h, 1175
CFE_PLATFORM_ES_MAX_BLOCK_SIZE
cpu1_platform_cfg.h, 1175
CFE_PLATFORM_ES_MAX_GEN_COUNTERS
cpu1_platform_cfg.h, 1176
CFE_PLATFORM_ES_MAX_LIBRARIES
cpu1_platform_cfg.h, 1176
CFE_PLATFORM_ES_MAX_MEMORY_POOLS
cpu1_platform_cfg.h, 1176
CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS
cpu1_platform_cfg.h, 1177
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
cpu1_platform_cfg.h, 1177
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02
cpu1_platform_cfg.h, 1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03
cpu1_platform_cfg.h, 1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04
cpu1_platform_cfg.h, 1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05
cpu1_platform_cfg.h, 1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11
cpu1_platform_cfg.h, 1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12

cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13
cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14
cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15
cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16
cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN
cpu1_platform_cfg.h, 1180
CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_STRING
cpu1_platform_cfg.h, 1181
CFE_PLATFORM_ES_NONVOL_STARTUP_FILE
cpu1_platform_cfg.h, 1181
CFE_PLATFORM_ES_OBJECT_TABLE_SIZE
cpu1_platform_cfg.h, 1181
CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY
cpu1_platform_cfg.h, 1182
CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
cpu1_platform_cfg.h, 1182
CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE
cpu1_platform_cfg.h, 1182
CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
cpu1_platform_cfg.h, 1183
CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
cpu1_platform_cfg.h, 1183
CFE_PLATFORM_ES_PERF_FILTMASK_ALL
cpu1_platform_cfg.h, 1183
CFE_PLATFORM_ES_PERF_FILTMASK_INIT
cpu1_platform_cfg.h, 1184
CFE_PLATFORM_ES_PERF_FILTMASK_NONE
cpu1_platform_cfg.h, 1184
CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
cpu1_platform_cfg.h, 1184
CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
cpu1_platform_cfg.h, 1185
CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
cpu1_platform_cfg.h, 1185
CFE_PLATFORM_ES_POOL_MAX_BUCKETS
cpu1_platform_cfg.h, 1185
CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
cpu1_platform_cfg.h, 1186
CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
cpu1_platform_cfg.h, 1186
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED
cpu1_platform_cfg.h, 1186
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
cpu1_platform_cfg.h, 1187
CFE_PLATFORM_ES_RESET_AREA_SIZE
cpu1_platform_cfg.h, 1187
CFE_PLATFORM_ES_START_TASK_PRIORITY
cpu1_platform_cfg.h, 1188
CFE_PLATFORM_ES_START_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1188
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC
cpu1_platform_cfg.h, 1189
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
cpu1_platform_cfg.h, 1189
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
cpu1_platform_cfg.h, 1190
CFE_PLATFORM_ES_USER_RESERVED_SIZE
cpu1_platform_cfg.h, 1190
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
cpu1_platform_cfg.h, 1191
CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
cpu1_platform_cfg.h, 1191
CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
cpu1_platform_cfg.h, 1192
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
cpu1_platform_cfg.h, 1192
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE
cpu1_platform_cfg.h, 1192
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
cpu1_platform_cfg.h, 1193
CFE_PLATFORM_EVS_LOG_MAX
cpu1_platform_cfg.h, 1193
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
cpu1_platform_cfg.h, 1193
CFE_PLATFORM_EVS_PORT_DEFAULT
cpu1_platform_cfg.h, 1194
CFE_PLATFORM_EVS_START_TASK_PRIORITY
cpu1_platform_cfg.h, 1194
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1194
CFE_PLATFORM_SB_BUF_MEMORY_BYTES
cpu1_platform_cfg.h, 1195
CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
cpu1_platform_cfg.h, 1195
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
cpu1_platform_cfg.h, 1196
CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
cpu1_platform_cfg.h, 1196
CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
cpu1_platform_cfg.h, 1197
CFE_PLATFORM_SB_FILTER_MASK1
cpu1_platform_cfg.h, 1197
CFE_PLATFORM_SB_FILTER_MASK2
cpu1_platform_cfg.h, 1197
CFE_PLATFORM_SB_FILTER_MASK3
cpu1_platform_cfg.h, 1198
CFE_PLATFORM_SB_FILTER_MASK4
cpu1_platform_cfg.h, 1198

CFE_PLATFORM_SB_FILTER_MASK5
cpu1_platform_cfg.h, 1198
CFE_PLATFORM_SB_FILTER_MASK6
cpu1_platform_cfg.h, 1198
CFE_PLATFORM_SB_FILTER_MASK7
cpu1_platform_cfg.h, 1198
CFE_PLATFORM_SB_FILTER_MASK8
cpu1_platform_cfg.h, 1198
CFE_PLATFORM_SB_FILTERED_EVENT1
cpu1_platform_cfg.h, 1199
CFE_PLATFORM_SB_FILTERED_EVENT2
cpu1_platform_cfg.h, 1199
CFE_PLATFORM_SB_FILTERED_EVENT3
cpu1_platform_cfg.h, 1199
CFE_PLATFORM_SB_FILTERED_EVENT4
cpu1_platform_cfg.h, 1199
CFE_PLATFORM_SB_FILTERED_EVENT5
cpu1_platform_cfg.h, 1199
CFE_PLATFORM_SB_FILTERED_EVENT6
cpu1_platform_cfg.h, 1200
CFE_PLATFORM_SB_FILTERED_EVENT7
cpu1_platform_cfg.h, 1200
CFE_PLATFORM_SB_FILTERED_EVENT8
cpu1_platform_cfg.h, 1200
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
cpu1_platform_cfg.h, 1200
CFE_PLATFORM_SB_MAX_BLOCK_SIZE
cpu1_platform_cfg.h, 1200
CFE_PLATFORM_SB_MAX_DEST_PER_PKT
cpu1_platform_cfg.h, 1201
CFE_PLATFORM_SB_MAX_MSG_IDS
cpu1_platform_cfg.h, 1201
CFE_PLATFORM_SB_MAX_PIPES
cpu1_platform_cfg.h, 1201
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
cpu1_platform_cfg.h, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02
cpu1_platform_cfg.h, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03
cpu1_platform_cfg.h, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09
cpu1_platform_cfg.h, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15
cpu1_platform_cfg.h, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16
cpu1_platform_cfg.h, 1205
CFE_PLATFORM_SB_START_TASK_PRIORITY
cpu1_platform_cfg.h, 1205
CFE_PLATFORM_SB_START_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1205
CFE_PLATFORM_TBL_BUF_MEMORY_BYTES
cpu1_platform_cfg.h, 1205
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE
cpu1_platform_cfg.h, 1206
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
cpu1_platform_cfg.h, 1206
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
cpu1_platform_cfg.h, 1206
CFE_PLATFORM_TBL_MAX_NUM_HANDLES
cpu1_platform_cfg.h, 1207
CFE_PLATFORM_TBL_MAX_NUM_TABLES
cpu1_platform_cfg.h, 1207
CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
cpu1_platform_cfg.h, 1207
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
cpu1_platform_cfg.h, 1208
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
cpu1_platform_cfg.h, 1208
CFE_PLATFORM_TBL_START_TASK_PRIORITY
cpu1_platform_cfg.h, 1209
CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1209
CFE_PLATFORM_TBL_U32FROM4CHARS
cpu1_platform_cfg.h, 1209
CFE_PLATFORM_TBL_VALID_PRID_1
cpu1_platform_cfg.h, 1210
CFE_PLATFORM_TBL_VALID_PRID_2
cpu1_platform_cfg.h, 1210
CFE_PLATFORM_TBL_VALID_PRID_3
cpu1_platform_cfg.h, 1210
CFE_PLATFORM_TBL_VALID_PRID_4
cpu1_platform_cfg.h, 1210
CFE_PLATFORM_TBL_VALID_PRID_COUNT
cpu1_platform_cfg.h, 1211
CFE_PLATFORM_TBL_VALID_SCID_1
cpu1_platform_cfg.h, 1211
CFE_PLATFORM_TBL_VALID_SCID_2
cpu1_platform_cfg.h, 1211

CFE_PLATFORM_TBL_VALID_SCID_COUNT
cpu1_platform_cfg.h, 1212
CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY
cpu1_platform_cfg.h, 1212
CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1212
CFE_PLATFORM_TIME_CFG_CLIENT
cpu1_platform_cfg.h, 1212
CFE_PLATFORM_TIME_CFG_LATCH_FLY
cpu1_platform_cfg.h, 1212
CFE_PLATFORM_TIME_CFG_SERVER
cpu1_platform_cfg.h, 1213
CFE_PLATFORM_TIME_CFG_SIGNAL
cpu1_platform_cfg.h, 1213
CFE_PLATFORM_TIME_CFG_SOURCE
cpu1_platform_cfg.h, 1213
CFE_PLATFORM_TIME_CFG_SRC_GPS
cpu1_platform_cfg.h, 1214
CFE_PLATFORM_TIME_CFG_SRC_MET
cpu1_platform_cfg.h, 1214
CFE_PLATFORM_TIME_CFG_SRC_TIME
cpu1_platform_cfg.h, 1214
CFE_PLATFORM_TIME_CFG_START_FLY
cpu1_platform_cfg.h, 1215
CFE_PLATFORM_TIME_CFG_TONE_LIMIT
cpu1_platform_cfg.h, 1215
CFE_PLATFORM_TIME_CFG_VIRTUAL
cpu1_platform_cfg.h, 1215
CFE_PLATFORM_TIME_MAX_DELTA_SECS
cpu1_platform_cfg.h, 1216
CFE_PLATFORM_TIME_MAX_DELTA_SUBS
cpu1_platform_cfg.h, 1216
CFE_PLATFORM_TIME_MAX_LOCAL_SECS
cpu1_platform_cfg.h, 1217
CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
cpu1_platform_cfg.h, 1217
CFE_PLATFORM_TIME_START_TASK_PRIORITY
cpu1_platform_cfg.h, 1217
CFE_PLATFORM_TIME_START_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1217
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY
cpu1_platform_cfg.h, 1218
CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE
cpu1_platform_cfg.h, 1218
CFE_PLATFORM_TLM_MID_BASE
cpu1_msgids.h, 1157
CFE_PSP_AttachExceptions
cfe_psp.h, 1658
CFE_PSP_Decompress
cfe_psp.h, 1658
CFE_PSP_ERROR_ADDRESS_MISALIGNED
cfe_psp.h, 1649
CFE_PSP_ERROR_NOT_IMPLEMENTED
cfe_psp.h, 1649
CFE_PSP_ERROR_TIMEOUT
cfe_psp.h, 1649
CFE_PSP_ERROR
cfe_psp.h, 1649
CFE_PSP_EepromPowerDown
cfe_psp.h, 1658
CFE_PSP_EepromPowerUp
cfe_psp.h, 1658
CFE_PSP_EepromWrite16
cfe_psp.h, 1658
CFE_PSP_EepromWrite32
cfe_psp.h, 1659
CFE_PSP_EepromWrite8
cfe_psp.h, 1659
CFE_PSP_EepromWriteDisable
cfe_psp.h, 1659
CFE_PSP_EepromWriteEnable
cfe_psp.h, 1659
CFE_PSP_Exception_CopyContext
cfe_psp.h, 1659
CFE_PSP_Exception_GetCount
cfe_psp.h, 1659
CFE_PSP_Exception_GetSummary
cfe_psp.h, 1660
CFE_PSP_FlushCaches
cfe_psp.h, 1660
CFE_PSP_Get_Dec
cfe_psp.h, 1660
CFE_PSP_Get_Timebase
cfe_psp.h, 1660
CFE_PSP_Get_Timer_Tick
cfe_psp.h, 1661
CFE_PSP_GetBuildNumber
cfe_psp.h, 1661
CFE_PSP_GetCDSSize
cfe_psp.h, 1661
CFE_PSP_GetCFETextSegmentInfo
cfe_psp.h, 1661
CFE_PSP_GetKernelTextSegmentInfo
cfe_psp.h, 1661
CFE_PSP_GetProcessorId
cfe_psp.h, 1662
CFE_PSP_GetProcessorName
cfe_psp.h, 1662
CFE_PSP_GetResetArea
cfe_psp.h, 1662
CFE_PSP_GetRestartType
cfe_psp.h, 1662
CFE_PSP_GetSpacecraftId
cfe_psp.h, 1662
CFE_PSP_GetTime
cfe_psp.h, 1662
CFE_PSP_GetTimerLow32Rollover
cfe_psp.h, 1663

CFE_PSP_GetTimerTicksPerSecond
cfe_psp.h, 1663
CFE_PSP_GetUserReservedArea
cfe_psp.h, 1663
CFE_PSP_GetVersionCodeName
cfe_psp.h, 1663
CFE_PSP_GetVersionNumber
cfe_psp.h, 1664
CFE_PSP_GetVersionString
cfe_psp.h, 1664
CFE_PSP_GetVolatileDiskMem
cfe_psp.h, 1664
CFE_PSP_INVALID_INT_NUM
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_ADDR
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_ATTR
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_RANGE
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_SIZE
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_TYPE
cfe_psp.h, 1650
CFE_PSP_INVALID_MEM_WORDSIZE
cfe_psp.h, 1651
CFE_PSP_INVALID_MODULE_ID
cfe_psp.h, 1651
CFE_PSP_INVALID_MODULE_NAME
cfe_psp.h, 1651
CFE_PSP_INVALID_POINTER
cfe_psp.h, 1651
CFE_PSP_InitSSR
cfe_psp.h, 1665
CFE_PSP_MEM_ANY
cfe_psp.h, 1651
CFE_PSP_MEM_ATTR_READWRITE
cfe_psp.h, 1652
CFE_PSP_MEM_ATTR_READ
cfe_psp.h, 1651
CFE_PSP_MEM_ATTR_WRITE
cfe_psp.h, 1652
CFE_PSP_MEM_EEPROM
cfe_psp.h, 1652
CFE_PSP_MEM_INVALID
cfe_psp.h, 1652
CFE_PSP_MEM_RAM
cfe_psp.h, 1652
CFE_PSP_MEM_SIZE_BYTE
cfe_psp.h, 1652
CFE_PSP_MEM_SIZE_DWORD
cfe_psp.h, 1653
CFE_PSP_MEM_SIZE_WORD
cfe_psp.h, 1653
CFE_PSP_Main
cfe_psp.h, 1665
CFE_PSP_MemCpy
cfe_psp.h, 1665
CFE_PSP_MemRangeGet
cfe_psp.h, 1665
CFE_PSP_MemRangeSet
cfe_psp.h, 1665
CFE_PSP_MemRanges
cfe_psp.h, 1665
CFE_PSP_MemRead16
cfe_psp.h, 1666
CFE_PSP_MemRead32
cfe_psp.h, 1666
CFE_PSP_MemRead8
cfe_psp.h, 1666
CFE_PSP_MemSet
cfe_psp.h, 1666
CFE_PSP_MemValidateRange
cfe_psp.h, 1666
CFE_PSP_MemWrite16
cfe_psp.h, 1667
CFE_PSP_MemWrite32
cfe_psp.h, 1667
CFE_PSP_MemWrite8
cfe_psp.h, 1667
CFE_PSP_NO_EXCEPTION_DATA
cfe_psp.h, 1653
CFE_PSP_PANIC_CORE_APP
cfe_psp.h, 1653
CFE_PSP_PANIC_GENERAL_FAILURE
cfe_psp.h, 1653
CFE_PSP_PANIC_MEMORY_ALLOC
cfe_psp.h, 1653
CFE_PSP_PANIC_NONVOL_DISK
cfe_psp.h, 1654
CFE_PSP_PANIC_STARTUP_SEM
cfe_psp.h, 1654
CFE_PSP_PANIC_STARTUP
cfe_psp.h, 1654
CFE_PSP_PANIC_VOLATILE_DISK
cfe_psp.h, 1654
CFE_PSP_Panic
cfe_psp.h, 1667
CFE_PSP_PortRead16
cfe_psp.h, 1667
CFE_PSP_PortRead32
cfe_psp.h, 1667
CFE_PSP_PortRead8
cfe_psp.h, 1668
CFE_PSP_PortWrite16
cfe_psp.h, 1668
CFE_PSP_PortWrite32
cfe_psp.h, 1668

CFE_PSP_PortWrite8
cfe_psp.h, 1668
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET
cfe_psp.h, 1654
CFE_PSP_RST_SUBTYPE_EXCEPTION
cfe_psp.h, 1654
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND
cfe_psp.h, 1655
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG
cfe_psp.h, 1655
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET
cfe_psp.h, 1655
CFE_PSP_RST_SUBTYPE_MAX
cfe_psp.h, 1655
CFE_PSP_RST_SUBTYPE_POWER_CYCLE
cfe_psp.h, 1655
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON
cfe_psp.h, 1656
CFE_PSP_RST_SUBTYPE_RESET_COMMAND
cfe_psp.h, 1656
CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET
cfe_psp.h, 1656
CFE_PSP_RST_TYPE_MAX
cfe_psp.h, 1656
CFE_PSP_RST_TYPE_POWERON
cfe_psp.h, 1656
CFE_PSP_RST_TYPE_PROCESSOR
cfe_psp.h, 1657
CFE_PSP_ReadFromCDS
cfe_psp.h, 1668
CFE_PSP_Restart
cfe_psp.h, 1668
CFE_PSP_SOFT_TIMEBASE_NAME
cfe_psp.h, 1657
CFE_PSP_SUCCESS
cfe_psp.h, 1657
CFE_PSP_SetDefaultExceptionEnvironment
cfe_psp.h, 1669
CFE_PSP_WatchdogDisable
cfe_psp.h, 1669
CFE_PSP_WatchdogEnable
cfe_psp.h, 1669
CFE_PSP_WatchdogGet
cfe_psp.h, 1669
CFE_PSP_WatchdogInit
cfe_psp.h, 1669
CFE_PSP_WatchdogService
cfe_psp.h, 1669
CFE_PSP_WatchdogSet
cfe_psp.h, 1669
CFE_PSP_WriteToCDS
cfe_psp.h, 1669
CFE_RESOURCEID_MAKE_BASE
cfe_resourceid_basevalue.h, 1467
CFE_RESOURCEID_MAX
cfe_resourceid_basevalue.h, 1467
CFE_RESOURCEID_RESERVED
cfe_resourceid_api_typedefs.h, 1316
CFE_RESOURCEID_SHIFT
cfe_resourceid_basevalue.h, 1467
CFE_RESOURCEID_TEST_DEFINED
cfe_resourceid.h, 1309
CFE_RESOURCEID_TEST_EQUAL
cfe_resourceid.h, 1309
CFE_RESOURCEID_TO_ULONG
cfe_resourceid.h, 1309
CFE_RESOURCEID_UNDEFINED
cfe_resourceid_api_typedefs.h, 1316
CFE_REVISION
cfe_version.h, 1347
CFE_ResourceId_Equal
cfe_resourceid.h, 1310
CFE_ResourceId_FindNext
cfe_resourceid.h, 1310
CFE_ResourceId_FromInteger
cfe_resourceid.h, 1311
CFE_ResourceId_GetBase
cfe_resourceid.h, 1312
CFE_ResourceId_GetSerial
cfe_resourceid.h, 1312
CFE_ResourceId_IsDefined
cfe_resourceid.h, 1313
CFE_ResourceId_ToIndex
cfe_resourceid.h, 1314
CFE_ResourceId_ToInteger
cfe_resourceid.h, 1315
CFE_SB_ALLSUBS_TLM_MID
cpu1_msgids.h, 1157
CFE_SB_AllSubscriptionsTlm, 764
Payload, 765
TelemetryHeader, 765
CFE_SB_AllSubscriptionsTlm_Payload, 765
Entries, 766
Entry, 766
PktSegment, 766
TotalSegments, 766
CFE_SB_AllSubscriptionsTlm_Payload_t
cfe_sb_msg.h, 1505
CFE_SB_AllSubscriptionsTlm_t
cfe_sb_msg.h, 1505
CFE_SB_AllocateMessageBuffer
cFE Zero Copy APIs, 460
CFE_SB_BAD_ARGUMENT
cFE Return Code Defines, 320
CFE_SB_BAD_CMD_CODE_EID
cfe_sb_events.h, 1470
CFE_SB_BAD_MSGID_EID
cfe_sb_events.h, 1470

CFE_SB_BAD_PIPEID_EID
cfe_sb_events.h, 1471
CFE_SB_BUF_ALOC_ERR
cFE Return Code Defines, 320
CFE_SB_BUFFER_INVALID
cFE Return Code Defines, 320
CFE_SB_Buffer_t
cfe_sb_api_typedefs.h, 1324
CFE_SB_CMD0_RCVD_EID
cfe_sb_events.h, 1471
CFE_SB_CMD1_RCVD_EID
cfe_sb_events.h, 1471
CFE_SB_CMD_MID
cpu1_msgids.h, 1157
CFE_SB_CR_PIPE_BAD_ARG_EID
cfe_sb_events.h, 1472
CFE_SB_CR_PIPE_ERR_EID
cfe_sb_events.h, 1472
CFE_SB_CR_PIPE_NAME_TAKEN_EID
cfe_sb_events.h, 1472
CFE_SB_CR_PIPE_NO_FREE_EID
cfe_sb_events.h, 1473
CFE_SB_CreatePipe
cFE Pipe Management APIs, 444
CFE_SB_DEFAULT_QOS
cfe_sb_api_typedefs.h, 1321
CFE_SB_DEL_PIPE_ERR1_EID
cfe_sb_events.h, 1473
CFE_SB_DEL_PIPE_ERR2_EID
cfe_sb_events.h, 1473
CFE_SB_DEST_BLK_ERR_EID
cfe_sb_events.h, 1474
CFE_SB_DISABLE_ROUTE_CC
cfe_sb_msg.h, 1494
CFE_SB_DISABLE_SUB_REPORTING_CC
cfe_sb_msg.h, 1494
CFE_SB_DSBL_RTE1_EID
cfe_sb_events.h, 1474
CFE_SB_DSBL_RTE2_EID
cfe_sb_events.h, 1474
CFE_SB_DSBL_RTE3_EID
cfe_sb_events.h, 1475
CFE_SB_DUP_SUBSCRIP_EID
cfe_sb_events.h, 1475
CFE_SB_DeletePipe
cFE Pipe Management APIs, 445
CFE_SB_DisableRouteCmd_t
cfe_sb_msg.h, 1505
CFE_SB_DisableSubReportingCmd_t
cfe_sb_msg.h, 1505
CFE_SB_ENABLE_ROUTE_CC
cfe_sb_msg.h, 1495
CFE_SB_ENABLE_SUB_REPORTING_CC
cfe_sb_msg.h, 1496
CFE_SB_ENBL_RTE1_EID
cfe_sb_events.h, 1475
CFE_SB_ENBL_RTE2_EID
cfe_sb_events.h, 1476
CFE_SB_ENBL_RTE3_EID
cfe_sb_events.h, 1476
CFE_SB_EnableRouteCmd_t
cfe_sb_msg.h, 1505
CFE_SB_EnableSubReportingCmd_t
cfe_sb_msg.h, 1505
CFE_SB_FILEWRITE_ERR_EID
cfe_sb_events.h, 1476
CFE_SB_FULL_SUB_PKT_EID
cfe_sb_events.h, 1477
CFE_SB_GET_BUF_ERR_EID
cfe_sb_events.h, 1477
CFE_SB_GETPIPEIDBYNAME_EID
cfe_sb_events.h, 1477
CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID
cfe_sb_events.h, 1478
CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID
cfe_sb_events.h, 1478
CFE_SB_GETPIPENAME_EID
cfe_sb_events.h, 1478
CFE_SB_GETPIPENAME_ID_ERR_EID
cfe_sb_events.h, 1479
CFE_SB_GETPIPENAME_NULL_PTR_EID
cfe_sb_events.h, 1479
CFE_SB_GETPIPEOPTS_EID
cfe_sb_events.h, 1479
CFE_SB_GETPIPEOPTS_ID_ERR_EID
cfe_sb_events.h, 1480
CFE_SB_GETPIPEOPTS_PTR_ERR_EID
cfe_sb_events.h, 1480
CFE_SB_GetPipeIdByName
cFE Pipe Management APIs, 446
CFE_SB_GetPipeName
cFE Pipe Management APIs, 447
CFE_SB_GetPipeOpts
cFE Pipe Management APIs, 447
CFE_SB_GetUserData
cFE Message Characteristics APIs, 463
CFE_SB_GetUserDataLength
cFE Message Characteristics APIs, 464
CFE_SB_HASHCOLLISION_EID
cfe_sb_events.h, 1480
CFE_SB_HK_TLM_MID
cpu1_msgids.h, 1157
CFE_SB_HousekeepingTlm, 767
Payload, 767
TelemetryHeader, 767
CFE_SB_HousekeepingTlm_Payload, 768
CommandCounter, 769
CommandErrorCounter, 769

CreatePipeErrorCounter, 769
DuplicateSubscriptionsCounter, 769
GetPipeIdByNameErrorCounter, 770
InternalErrorCounter, 770
MemInUse, 770
MemPoolHandle, 770
MsgLimitErrorCounter, 771
MsgReceiveErrorCounter, 771
MsgSendErrorCounter, 771
NoSubscribersCounter, 771
PipeOptsErrorCounter, 772
PipeOverflowErrorCounter, 772
Spare2Align, 772
SubscribeErrorCounter, 772
UnmarkedMem, 773
CFE_SB_HousekeepingTlm_Payload_t
cfe_sb_msg.h, 1506
CFE_SB_HousekeepingTlm_t
cfe_sb_msg.h, 1506
CFE_SB_INIT_EID
cfe_sb_events.h, 1481
CFE_SB_INTERNAL_ERR
cFE Return Code Defines, 321
CFE_SB_INVALID_MSG_ID
cfe_sb_api_typedefs.h, 1321
CFE_SB_INVALID_PIPE
cfe_sb_api_typedefs.h, 1321
CFE_SB_IsValidMsgId
cFE Message ID APIs, 468
CFE_SB_LEN_ERR_EID
cfe_sb_events.h, 1481
CFE_SB_MAX_DESTS_MET_EID
cfe_sb_events.h, 1481
CFE_SB_MAX_DESTS_MET
cFE Return Code Defines, 321
CFE_SB_MAX_MSGS_MET_EID
cfe_sb_events.h, 1482
CFE_SB_MAX_MSGS_MET
cFE Return Code Defines, 321
CFE_SB_MAX_PIPES_MET_EID
cfe_sb_events.h, 1482
CFE_SB_MAX_PIPES_MET
cFE Return Code Defines, 321
CFE_SB_MSG_TOO_BIG_EID
cfe_sb_events.h, 1482
CFE_SB_MSG_TOO_BIG
cFE Return Code Defines, 322
CFE_SB_MSGID_LIM_ERR_EID
cfe_sb_events.h, 1483
CFE_SB_MSGID_RESERVED
cfe_sb_api_typedefs.h, 1322
CFE_SB_MSGID_UNWRAP_VALUE
cfe_sb_api_typedefs.h, 1322
CFE_SB_MSGID_WRAP_VALUE
cfe_sb_api_typedefs.h, 1322
CFE_SB_MSGID_C
cfe_sb_api_typedefs.h, 1321
CFE_SB_MessageStringGet
cFE Message Characteristics APIs, 464
CFE_SB_MessageStringSet
cFE Message Characteristics APIs, 465
CFE_SB_Msg, 773
LongDouble, 774
LongInt, 774
Msg, 774
CFE_SB_MsgId_Atom_t
cfe_sb_extern_typedefs.h, 1326
CFE_SB_MsgId_Equal
cFE Message ID APIs, 468
CFE_SB_MsgId_t, 774
Value, 775
CFE_SB_MsgIdToValue
cFE Message ID APIs, 469
CFE_SB_MsgMapFileEntry, 775
Index, 776
MsgId, 776
CFE_SB_MsgMapFileEntry_t
cfe_sb_msg.h, 1506
CFE_SB_NO_MESSAGE
cFE Return Code Defines, 322
CFE_SB_NOOP_CC
cfe_sb_msg.h, 1497
CFE_SB_NOT_IMPLEMENTED
cFE Return Code Defines, 322
CFE_SB_NoopCmd_t
cfe_sb_msg.h, 1506
CFE_SB_ONESUB_TLM_MID
cpu1_msgids.h, 1157
CFE_SB_PART_SUB_PKT_EID
cfe_sb_events.h, 1483
CFE_SB_PEND_FOREVER
cfe_sb_api_typedefs.h, 1323
CFE_SB_PIPE_ADDED_EID
cfe_sb_events.h, 1483
CFE_SB_PIPE_CR_ERR
cFE Return Code Defines, 322
CFE_SB_PIPE_DELETED_EID
cfe_sb_events.h, 1484
CFE_SB_PIPE_RD_ERR
cFE Return Code Defines, 323
CFE_SB_PIPEID_C
cfe_sb_api_typedefs.h, 1323
CFE_SB_PIPEOPTS_IGNOREMINE
cFE SB Pipe options, 471
CFE_SB_POLL
cfe_sb_api_typedefs.h, 1323
CFE_SB_PipeDepthStats, 776
CurrentQueueDepth, 777

MaxQueueDepth, 777
PeakQueueDepth, 777
PipeId, 778
Spare, 778
CFE_SB_PipeDepthStats_t
cfe_sb_msg.h, 1506
CFE_SB_PipeId_ToIndex
cFE Pipe Management APIs, 449
CFE_SB_PipeId_t
cfe_sb_extern_typedefs.h, 1326
CFE_SB_PipeInfoEntry, 778
AppId, 779
AppName, 779
CurrentQueueDepth, 779
MaxQueueDepth, 780
Opts, 780
PeakQueueDepth, 780
PipeId, 780
PipeName, 780
SendErrors, 781
Spare, 781
CFE_SB_PipeInfoEntry_t
cfe_sb_msg.h, 1506
CFE_SB_Q_FULL_ERR_EID
cfe_sb_events.h, 1484
CFE_SB_Q_RD_ERR_EID
cfe_sb_events.h, 1484
CFE_SB_Q_WR_ERR_EID
cfe_sb_events.h, 1485
CFE_SB_Qos_t, 781
Priority, 782
Reliability, 782
CFE_SB_QosPriority
cfe_sb_extern_typedefs.h, 1327
CFE_SB_QosPriority_Enum_t
cfe_sb_extern_typedefs.h, 1326
CFE_SB_QosReliability
cfe_sb_extern_typedefs.h, 1327
CFE_SB_QosReliability_Enum_t
cfe_sb_extern_typedefs.h, 1326
CFE_SB_RCV_BAD_ARG_EID
cfe_sb_events.h, 1485
CFE_SB_RESET_COUNTERS_CC
cfe_sb_msg.h, 1498
CFE_SB_ReceiveBuffer
cFE Send/Receive Message APIs, 457
CFE_SB_ReleaseMessageBuffer
cFE Zero Copy APIs, 461
CFE_SB_ResetCountersCmd_t
cfe_sb_msg.h, 1507
CFE_SB_RouteCmd, 782
CommandHeader, 783
Payload, 783
CFE_SB_RouteCmd_Payload, 783
MsgId, 784
Pipe, 784
Spare, 784
CFE_SB_RouteCmd_Payload_t
cfe_sb_msg.h, 1507
CFE_SB_RouteCmd_t
cfe_sb_msg.h, 1507
CFE_SB_RouteId_Atom_t
cfe_sb_extern_typedefs.h, 1327
CFE_SB_RoutingFileEntry, 785
AppName, 785
MsgCnt, 785
MsgId, 786
PipeId, 786
PipeName, 786
State, 786
CFE_SB_RoutingFileEntry_t
cfe_sb_msg.h, 1507
CFE_SB_SEND_BAD_ARG_EID
cfe_sb_events.h, 1485
CFE_SB_SEND_HK_MID
cpu1_msgids.h, 1158
CFE_SB_SEND_INV_MSGID_EID
cfe_sb_events.h, 1486
CFE_SB_SEND_NO_SUBS_EID
cfe_sb_events.h, 1486
CFE_SB_SEND_PREV_SUBS_CC
cfe_sb_msg.h, 1499
CFE_SB_SEND_SB_STATS_CC
cfe_sb_msg.h, 1500
CFE_SB_SETPIPEOPTS_EID
cfe_sb_events.h, 1486
CFE_SB_SETPIPEOPTS_ID_ERR_EID
cfe_sb_events.h, 1487
CFE_SB_SETPIPEOPTS_OWNER_ERR_EID
cfe_sb_events.h, 1487
CFE_SB_SND_RTG_EID
cfe_sb_events.h, 1487
CFE_SB_SND_RTG_ERR1_EID
cfe_sb_events.h, 1488
CFE_SB_SND_STATS_EID
cfe_sb_events.h, 1488
CFE_SB_STATS_TLM_MID
cpu1_msgids.h, 1158
CFE_SB_SUB_ARG_ERR_EID
cfe_sb_events.h, 1488
CFE_SB_SUB_ENTRIES_PER_PKT
cfe_sb_extern_typedefs.h, 1325
CFE_SB_SUB_INV_CALLER_EID
cfe_sb_events.h, 1489
CFE_SB_SUB_INV_PIPE_EID
cfe_sb_events.h, 1489
CFE_SB_SUB_RPT_CTRL_MID
cpu1_msgids.h, 1158

CFE_SB_SUBSCRIPTION_RCVD_EID
cfe_sb_events.h, 1489
CFE_SB_SUBSCRIPTION_REMOVED_EID
cfe_sb_events.h, 1490
CFE_SB_SUBSCRIPTION_RPT_EID
cfe_sb_events.h, 1490
CFE_SB_SUBSCRIPTION
cfe_sb_api_typedefs.h, 1324
CFE_SB_SendPrevSubsCmd_t
cfe_sb_msg.h, 1507
CFE_SB_SendSbStatsCmd_t
cfe_sb_msg.h, 1508
CFE_SB_SetPipeOpts
cFE Pipe Management APIs, 450
CFE_SB_SetUserDataLength
cFE Message Characteristics APIs, 466
CFE_SB_SingleSubscriptionTlm, 787
Payload, 787
TelemetryHeader, 787
CFE_SB_SingleSubscriptionTlm_Payload, 787
MsgId, 788
Pipe, 788
Qos, 788
SubType, 789
CFE_SB_SingleSubscriptionTlm_Payload_t
cfe_sb_msg.h, 1508
CFE_SB_SingleSubscriptionTlm_t
cfe_sb_msg.h, 1508
CFE_SB_StatsTlm, 789
Payload, 789
TelemetryHeader, 790
CFE_SB_StatsTlm_Payload, 790
MaxMemAllowed, 791
MaxMsgIdsAllowed, 791
MaxPipeDepthAllowed, 791
MaxPipesAllowed, 792
MaxSubscriptionsAllowed, 792
MemInUse, 792
MsgIdsInUse, 792
PeakMemInUse, 793
PeakMsgIdsInUse, 793
PeakPipesInUse, 793
PeakSBBuffersInUse, 793
PeakSubscriptionsInUse, 794
PipeDepthStats, 794
PipesInUse, 794
SBBuffersInUse, 794
SubscriptionsInUse, 795
CFE_SB_StatsTlm_Payload_t
cfe_sb_msg.h, 1508
CFE_SB_StatsTlm_t
cfe_sb_msg.h, 1508
CFE_SB_SubEntries, 795
MsgId, 796
Pipe, 796
Qos, 796
CFE_SB_SubEntries_t
cfe_sb_msg.h, 1509
CFE_SB_Subscribe
cFE Message Subscription Control APIs, 451
CFE_SB_SubscribeEx
cFE Message Subscription Control APIs, 452
CFE_SB_SubscribeLocal
cFE Message Subscription Control APIs, 453
CFE_SB_TIME_OUT
cFE Return Code Defines, 323
CFE_SB_TimeStampMsg
cFE Message Characteristics APIs, 467
CFE_SB_TransmitBuffer
cFE Zero Copy APIs, 461
CFE_SB_TransmitMsg
cFE Send/Receive Message APIs, 458
CFE_SB_UNSUB_ARG_ERR_EID
cfe_sb_events.h, 1490
CFE_SB_UNSUB_INV_CALLER_EID
cfe_sb_events.h, 1491
CFE_SB_UNSUB_INV_PIPE_EID
cfe_sb_events.h, 1491
CFE_SB_UNSUB_NO_SUBS_EID
cfe_sb_events.h, 1491
CFE_SB_UNSUBSCRIPTION
cfe_sb_api_typedefs.h, 1324
CFE_SB_Unsubscribe
cFE Message Subscription Control APIs, 454
CFE_SB_UnsubscribeLocal
cFE Message Subscription Control APIs, 455
CFE_SB_ValueToMsgId
cFE Message ID APIs, 469
CFE_SB_WRITE_MAP_INFO_CC
cfe_sb_msg.h, 1501
CFE_SB_WRITE_PIPE_INFO_CC
cfe_sb_msg.h, 1502
CFE_SB_WRITE_ROUTING_INFO_CC
cfe_sb_msg.h, 1503
CFE_SB_WRONG_MSG_TYPE
cFE Return Code Defines, 323
CFE_SB_WriteFileInfoCmd, 797
CommandHeader, 797
Payload, 797
CFE_SB_WriteFileInfoCmd_Payload, 798
Filename, 798
CFE_SB_WriteFileInfoCmd_Payload_t
cfe_sb_msg.h, 1509
CFE_SB_WriteFileInfoCmd_t
cfe_sb_msg.h, 1509
CFE_SB_WriteMapInfoCmd_t
cfe_sb_msg.h, 1509
CFE_SB_WritePipeInfoCmd_t

cfe_sb_msg.h, 1509
CFE_SB_WriteRoutingInfoCmd_t
cfe_sb_msg.h, 1509
CFE_SERVICE_BITMASK
cfe_error.h, 1256
CFE_SEVERITY_BITMASK
cfe_error.h, 1256
CFE_SEVERITY_ERROR
cfe_error.h, 1256
CFE_SEVERITY_INFO
cfe_error.h, 1257
CFE_SEVERITY_SUCCESS
cfe_error.h, 1257
CFE_SET
cfe_sb.h, 1319
CFE_SOFTWARE_BUS_SERVICE
cfe_error.h, 1257
CFE_SRC_VERSION
cfe_version.h, 1348
CFE_STATUS_BAD_COMMAND_CODE
cFE Return Code Defines, 323
CFE_STATUS_EXTERNAL_RESOURCE_FAIL
cFE Return Code Defines, 324
CFE_STATUS_NO_COUNTER_INCREMENT
cFE Return Code Defines, 324
CFE_STATUS_NOT_IMPLEMENTED
cFE Return Code Defines, 324
CFE_STATUS_REQUEST_ALREADY_PENDING
cFE Return Code Defines, 324
CFE_STATUS_UNKNOWN_MSG_ID
cFE Return Code Defines, 325
CFE_STATUS_WRONG_MSG_LENGTH
cFE Return Code Defines, 325
CFE_STR_HELPER
cfe_version.h, 1348
CFE_STR
cfe_version.h, 1348
CFE_SUCCESS
cFE Return Code Defines, 325
CFE_Status_t
cfe_error.h, 1258
CFE_TABLE_SERVICE
cfe_error.h, 1257
CFE_TBL_ABORT_LOAD_CC
cfe_tbl_msg.h, 1538
CFE_TBL_ACTIVATE_CC
cfe_tbl_msg.h, 1539
CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID
cfe_tbl_events.h, 1512
CFE_TBL_ACTIVATE_ERR_EID
cfe_tbl_events.h, 1513
CFE_TBL_ASSUMED_VALID_INF_EID
cfe_tbl_events.h, 1513
CFE_TBL_AbortLoadCmd, 798
CommandHeader, 799
Payload, 799
CFE_TBL_AbortLoadCmd_Payload, 799
TableName, 800
CFE_TBL_AbortLoadCmd_Payload_t
cfe_tbl_msg.h, 1549
CFE_TBL_AbortLoadCmd_t
cfe_tbl_msg.h, 1549
CFE_TBL_ActivateCmd, 800
CommandHeader, 801
Payload, 801
CFE_TBL_ActivateCmd_Payload, 801
TableName, 802
CFE_TBL_ActivateCmd_Payload_t
cfe_tbl_msg.h, 1549
CFE_TBL_ActivateCmd_t
cfe_tbl_msg.h, 1549
CFE_TBL_BAD_ARGUMENT
cFE Return Code Defines, 325
CFE_TBL_BAD_TABLE_HANDLE
cfe_tbl_api_typedefs.h, 1331
CFE_TBL_BufferSelect
cfe_tbl_extern_typedefs.h, 1333
CFE_TBL_BufferSelect_Enum_t
cfe_tbl_extern_typedefs.h, 1333
CFE_TBL_CC1_ERR_EID
cfe_tbl_events.h, 1513
CFE_TBL_CDS_DELETE_ERR_EID
cfe_tbl_events.h, 1514
CFE_TBL_CDS_DELETED_INFO_EID
cfe_tbl_events.h, 1514
CFE_TBL_CDS_NOT_FOUND_ERR_EID
cfe_tbl_events.h, 1514
CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID
cfe_tbl_events.h, 1515
CFE_TBL_CMD_MID
cpu1_msgids.h, 1158
CFE_TBL_CREATING_DUMP_FILE_ERR_EID
cfe_tbl_events.h, 1515
CFE_TBL_CallbackFuncPtr_t
cfe_tbl_api_typedefs.h, 1331
CFE_TBL_DELETE_CDS_CC
cfe_tbl_msg.h, 1540
CFE_TBL_DUMP_CC
cfe_tbl_msg.h, 1541
CFE_TBL_DUMP_PENDING_ERR_EID
cfe_tbl_events.h, 1515
CFE_TBL_DUMP_REGISTRY_CC
cfe_tbl_msg.h, 1542
CFE_TBL_DelCDSCmd_Payload, 802
TableName, 803
CFE_TBL_DelCDSCmd_Payload_t
cfe_tbl_msg.h, 1549
CFE_TBL_DeleteCDSCmd, 803

CommandHeader, 803
Payload, 804
CFE_TBL_DeleteCDSCmd_t
cfe_tbl_msg.h, 1549
CFE_TBL_DumpCmd, 804
CommandHeader, 804
Payload, 805
CFE_TBL_DumpCmd_Payload, 805
ActiveTableFlag, 806
DumpFilename, 806
TableName, 806
CFE_TBL_DumpCmd_Payload_t
cfe_tbl_msg.h, 1550
CFE_TBL_DumpCmd_t
cfe_tbl_msg.h, 1550
CFE_TBL_DumpRegistryCmd, 806
CommandHeader, 807
Payload, 807
CFE_TBL_DumpRegistryCmd_Payload, 807
DumpFilename, 808
CFE_TBL_DumpRegistryCmd_Payload_t
cfe_tbl_msg.h, 1550
CFE_TBL_DumpRegistryCmd_t
cfe_tbl_msg.h, 1550
CFE_TBL_DumpToBuffer
cFE Manage Table Content APIs, 478
CFE_TBL_ERR_ACCESS
cFE Return Code Defines, 326
CFE_TBL_ERR_BAD_CONTENT_ID
cFE Return Code Defines, 326
CFE_TBL_ERR_BAD_PROCESSOR_ID
cFE Return Code Defines, 326
CFE_TBL_ERR_BAD_SPACECRAFT_ID
cFE Return Code Defines, 326
CFE_TBL_ERR_BAD_SUBTYPE_ID
cFE Return Code Defines, 327
CFE_TBL_ERR_DUMP_ONLY
cFE Return Code Defines, 327
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE
cFE Return Code Defines, 327
CFE_TBL_ERR_DUPLICATE_NOT_OWNED
cFE Return Code Defines, 327
CFE_TBL_ERR_FILE_FOR_WRONG_TABLE
cFE Return Code Defines, 328
CFE_TBL_ERR_FILE_SIZE_INCONSISTENT
cFE Return Code Defines, 328
CFE_TBL_ERR_FILE_TOO_LARGE
cFE Return Code Defines, 328
CFE_TBL_ERR_FILENAME_TOO_LONG
cFE Return Code Defines, 328
CFE_TBL_ERR_HANDLES_FULL
cFE Return Code Defines, 329
CFE_TBL_ERR_ILLEGAL_SRC_TYPE
cFE Return Code Defines, 329
CFE_TBL_ERR_INVALID_HANDLE
cFE Return Code Defines, 329
CFE_TBL_ERR_INVALID_NAME
cFE Return Code Defines, 329
CFE_TBL_ERR_INVALID_OPTIONS
cFE Return Code Defines, 330
CFE_TBL_ERR_INVALID_SIZE
cFE Return Code Defines, 330
CFE_TBL_ERR_LOAD_IN_PROGRESS
cFE Return Code Defines, 330
CFE_TBL_ERR_LOAD_INCOMPLETE
cFE Return Code Defines, 331
CFE_TBL_ERR_NEVER_LOADED
cFE Return Code Defines, 331
CFE_TBL_ERR_NO_ACCESS
cFE Return Code Defines, 331
CFE_TBL_ERR_NO_BUFFER_AVAIL
cFE Return Code Defines, 331
CFE_TBL_ERR_NO_STD_HEADER
cFE Return Code Defines, 332
CFE_TBL_ERR_NO_TBL_HEADER
cFE Return Code Defines, 332
CFE_TBL_ERR_PARTIAL_LOAD
cFE Return Code Defines, 332
CFE_TBL_ERR_REGISTRY_FULL
cFE Return Code Defines, 332
CFE_TBL_ERR_SHORT_FILE
cFE Return Code Defines, 333
CFE_TBL_ERR_UNREGISTERED
cFE Return Code Defines, 333
CFE_TBL_FAIL_HK_SEND_ERR_EID
cfe_tbl_events.h, 1516
CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID
cfe_tbl_events.h, 1516
CFE_TBL_FILE_ACCESS_ERR_EID
cfe_tbl_events.h, 1516
CFE_TBL_FILE_INCOMPLETE_ERR_EID
cfe_tbl_events.h, 1517
CFE_TBL_FILE_LOADED_INF_EID
cfe_tbl_events.h, 1517
CFE_TBL_FILE_STD_HDR_ERR_EID
cfe_tbl_events.h, 1517
CFE_TBL_FILE_SUBTYPE_ERR_EID
cfe_tbl_events.h, 1518
CFE_TBL_FILE_TBL_HDR_ERR_EID
cfe_tbl_events.h, 1518
CFE_TBL_FILE_TOO_BIG_ERR_EID
cfe_tbl_events.h, 1518
CFE_TBL_FILE_TYPE_ERR_EID
cfe_tbl_events.h, 1519
CFE_TBL_FILEDEF
cfe_tbl_filedef.h, 1335
CFE_TBL_File_Hdr, 808
NumBytes, 809

Offset, 809
Reserved, 809
TableName, 809
CFE_TBL_File_Hdr_t
cfe_tbl_extern_typedefs.h, 1333
CFE_TBL_FileDef, 810
Description, 810
ObjectName, 810
ObjectSize, 810
TableName, 811
TgtFilename, 811
CFE_TBL_FileDef_t
cfe_tbl_filedef.h, 1335
CFE_TBL_GetAddress
cFE Access Table Content APIs, 485
CFE_TBL_GetAddresses
cFE Access Table Content APIs, 486
CFE_TBL_GetInfo
cFE Get Table Information APIs, 490
CFE_TBL_GetStatus
cFE Get Table Information APIs, 491
CFE_TBL_HANDLE_ACCESS_ERR_EID
cfe_tbl_events.h, 1519
CFE_TBL_HK_TLM_MID
cpu1_msgids.h, 1158
CFE_TBL_Handle_t
cfe_tbl_api_typedefs.h, 1331
CFE_TBL_HousekeepingTlm, 811
Payload, 812
TelemetryHeader, 812
CFE_TBL_HousekeepingTlm_Payload, 812
ActiveBuffer, 813
ByteAlignPad1, 813
CommandCounter, 814
CommandErrorCounter, 814
FailedValCounter, 814
LastFileDumped, 814
LastFileLoaded, 815
LastTableLoaded, 815
LastUpdateTime, 815
LastUpdatedTable, 815
LastValCrc, 816
LastValStatus, 816
LastValTableName, 816
MemPoolHandle, 816
NumFreeSharedBufs, 817
NumLoadPending, 817
NumTables, 817
NumValRequests, 817
SuccessValCounter, 818
ValidationCounter, 818
CFE_TBL_HousekeepingTlm_Payload_t
cfe_tbl_msg.h, 1550
CFE_TBL_HousekeepingTlm_t
cfe_tbl_msg.h, 1550
CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID
cfe_tbl_events.h, 1519
CFE_TBL_IN_REGISTRY_ERR_EID
cfe_tbl_events.h, 1520
CFE_TBL_INFO_DUMP_PENDING
cFE Return Code Defines, 333
CFE_TBL_INFO_NO_UPDATE_PENDING
cFE Return Code Defines, 333
CFE_TBL_INFO_NO_VALIDATION_PENDING
cFE Return Code Defines, 334
CFE_TBL_INFO_RECOVERED_TBL
cFE Return Code Defines, 334
CFE_TBL_INFO_TABLE_LOCKED
cFE Return Code Defines, 334
CFE_TBL_INFO_UPDATE_PENDING
cFE Return Code Defines, 334
CFE_TBL_INFO_UPDATED
cFE Return Code Defines, 335
CFE_TBL_INFO_VALIDATION_PENDING
cFE Return Code Defines, 335
CFE_TBL_INIT_INF_EID
cfe_tbl_events.h, 1520
CFE_TBL_INTERNAL_ERROR_ERR_EID
cfe_tbl_events.h, 1520
CFE_TBL_Info, 818
Crc, 819
Critical, 819
DoubleBuffered, 820
DumpOnly, 820
FileCreateTimeSecs, 820
FileCreateTimeSubSecs, 820
LastFileLoaded, 820
NumUsers, 821
Size, 821
TableLoadedOnce, 821
TimeOfLastUpdate, 821
UserDefAddr, 821
CFE_TBL_Info_t
cfe_tbl_api_typedefs.h, 1331
CFE_TBL_LEN_ERR_EID
cfe_tbl_events.h, 1521
CFE_TBL_LOAD_ABORT_ERR_EID
cfe_tbl_events.h, 1521
CFE_TBL_LOAD_ABORT_INF_EID
cfe_tbl_events.h, 1521
CFE_TBL_LOAD_CC
cfe_tbl_msg.h, 1543
CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID
cfe_tbl_events.h, 1522
CFE_TBL_LOAD_FILENAME_LONG_ERR_EID
cfe_tbl_events.h, 1522
CFE_TBL_LOAD_IN_PROGRESS_ERR_EID
cfe_tbl_events.h, 1522

CFE_TBL_LOAD_PEND_REQ_INF_EID
cfe_tbl_events.h, 1523
CFE_TBL_LOAD_SUCCESS_INF_EID
cfe_tbl_events.h, 1523
CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_EID
cfe_tbl_events.h, 1523
CFE_TBL_LOAD_TYPE_ERR_EID
cfe_tbl_events.h, 1524
CFE_TBL_LOAD_VAL_ERR_EID
cfe_tbl_events.h, 1524
CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID
cfe_tbl_events.h, 1524
CFE_TBL_LOADING_PENDING_ERR_EID
cfe_tbl_events.h, 1525
CFE_TBL_Load
cFE Manage Table Content APIs, 479
CFE_TBL_LoadCmd, 822
CommandHeader, 822
Payload, 822
CFE_TBL_LoadCmd_Payload, 823
LoadFilename, 823
CFE_TBL_LoadCmd_Payload_t
cfe_tbl_msg.h, 1551
CFE_TBL_LoadCmd_t
cfe_tbl_msg.h, 1551
CFE_TBL_MAX_FULL_NAME_LEN
cfe_tbl_api_typedefs.h, 1331
CFE_TBL_MESSAGE_ERROR
cFE Return Code Defines, 335
CFE_TBL_MID_ERR_EID
cfe_tbl_events.h, 1525
CFE_TBL_Manage
cFE Manage Table Content APIs, 480
CFE_TBL_Modified
cFE Manage Table Content APIs, 481
CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID
cfe_tbl_events.h, 1525
CFE_TBL_NO_SUCH_TABLE_ERR_EID
cfe_tbl_events.h, 1526
CFE_TBL_NO_WORK_BUFFERS_ERR_EID
cfe_tbl_events.h, 1526
CFE_TBL_NOOP_CC
cfe_tbl_msg.h, 1544
CFE_TBL_NOOP_INF_EID
cfe_tbl_events.h, 1526
CFE_TBL_NOT_CRITICAL_TBL_ERR_EID
cfe_tbl_events.h, 1527
CFE_TBL_NOT_IMPLEMENTED
cFE Return Code Defines, 335
CFE_TBL_NOT_IN_CRIT_REG_ERR_EID
cfe_tbl_events.h, 1527
CFE_TBL_NoArgsCmd, 824
CommandHeader, 824
CFE_TBL_NoArgsCmd_t
cfe_tbl_msg.h, 1551
CFE_TBL_NoopCmd_t
cfe_tbl_msg.h, 1551
CFE_TBL_NotifyByMessage
cFE Get Table Information APIs, 492
CFE_TBL_NotifyCmd, 824
CommandHeader, 825
Payload, 825
CFE_TBL_NotifyCmd_Payload, 825
Parameter, 826
CFE_TBL_NotifyCmd_Payload_t
cfe_tbl_msg.h, 1551
CFE_TBL_NotifyCmd_t
cfe_tbl_msg.h, 1552
CFE_TBL_OPT_BUFFER_MSK
cFE Table Type Defines, 494
CFE_TBL_OPT_CRITICAL_MSK
cFE Table Type Defines, 495
CFE_TBL_OPT_CRITICAL
cFE Table Type Defines, 494
CFE_TBL_OPT_DBL_BUFFER
cFE Table Type Defines, 495
CFE_TBL_OPT_DEFAULT
cFE Table Type Defines, 495
CFE_TBL_OPT_DUMP_ONLY
cFE Table Type Defines, 495
CFE_TBL_OPT_LD_DMP_MSK
cFE Table Type Defines, 495
CFE_TBL_OPT_LOAD_DUMP
cFE Table Type Defines, 496
CFE_TBL_OPT_NOT_CRITICAL
cFE Table Type Defines, 496
CFE_TBL_OPT_NOT_USR_DEF
cFE Table Type Defines, 496
CFE_TBL_OPT_SNGL_BUFFER
cFE Table Type Defines, 496
CFE_TBL_OPT_USR_DEF_ADDR
cFE Table Type Defines, 496
CFE_TBL_OPT_USR_DEF_MSK
cFE Table Type Defines, 497
CFE_TBL_OVERWRITE_DUMP_INF_EID
cfe_tbl_events.h, 1527
CFE_TBL_OVERWRITE_REG_DUMP_INF_EID
cfe_tbl_events.h, 1528
CFE_TBL_PARTIAL_LOAD_ERR_EID
cfe_tbl_events.h, 1528
CFE_TBL_PROCESSOR_ID_ERR_EID
cfe_tbl_events.h, 1528
CFE_TBL_REG_TLM_MID
cpu1_msgids.h, 1158
CFE_TBL_REGISTER_ERR_EID
cfe_tbl_events.h, 1529
CFE_TBL_RESET_COUNTERS_CC
cfe_tbl_msg.h, 1545

CFE_TBL_RESET_INF_EID
cfe_tbl_events.h, 1529
CFE_TBL_Register
cFE Registration APIs, 472
CFE_TBL_ReleaseAddress
cFE Access Table Content APIs, 487
CFE_TBL_ReleaseAddresses
cFE Access Table Content APIs, 488
CFE_TBL_ResetCountersCmd_t
cfe_tbl_msg.h, 1552
CFE_TBL_SEND_HK_MID
cpu1_msgids.h, 1159
CFE_TBL_SEND_REGISTRY_CC
cfe_tbl_msg.h, 1546
CFE_TBL_SHARE_ERR_EID
cfe_tbl_events.h, 1529
CFE_TBL_SPACECRAFT_ID_ERR_EID
cfe_tbl_events.h, 1530
CFE_TBL_SendRegistryCmd, 826
CommandHeader, 827
Payload, 827
CFE_TBL_SendRegistryCmd_Payload, 827
TableName, 828
CFE_TBL_SendRegistryCmd_Payload_t
cfe_tbl_msg.h, 1552
CFE_TBL_SendRegistryCmd_t
cfe_tbl_msg.h, 1552
CFE_TBL_Share
cFE Registration APIs, 475
CFE_TBL_SrcEnum
cfe_tbl_api_typedefs.h, 1332
CFE_TBL_SrcEnum_t
cfe_tbl_api_typedefs.h, 1332
CFE_TBL_TLM_REG_CMD_INF_EID
cfe_tbl_events.h, 1530
CFE_TBL_TOO_MANY_DUMPS_ERR_EID
cfe_tbl_events.h, 1530
CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID
cfe_tbl_events.h, 1531
CFE_TBL_TableRegistryTlm, 828
Payload, 829
TelemetryHeader, 829
CFE_TBL_TableRegistryTlm_t
cfe_tbl_msg.h, 1552
CFE_TBL_TblRegPacket_Payload, 829
ActiveBufferAddr, 830
ByteAlign4, 830
Crc, 831
Critical, 831
DoubleBuffered, 831
DumpOnly, 831
FileCreateTimeSecs, 832
FileCreateTimeSubSecs, 832
InactiveBufferAddr, 832
LastFileLoaded, 832
LoadPending, 833
Name, 833
OwnerAppName, 833
Size, 833
TableLoadedOnce, 834
TimeOfLastUpdate, 834
ValidationFuncPtr, 834
CFE_TBL_TblRegPacket_Payload_t
cfe_tbl_msg.h, 1552
CFE_TBL_UNREGISTER_ERR_EID
cfe_tbl_events.h, 1531
CFE_TBL_UNVALIDATED_ERR_EID
cfe_tbl_events.h, 1531
CFE_TBL_UPDATE_ERR_EID
cfe_tbl_events.h, 1532
CFE_TBL_UPDATE_SUCCESS_INF_EID
cfe_tbl_events.h, 1532
CFE_TBL_Unregister
cFE Registration APIs, 476
CFE_TBL_Update
cFE Manage Table Content APIs, 482
CFE_TBL_VAL_REQ_MADE_INF_EID
cfe_tbl_events.h, 1532
CFE_TBL_VALIDATE_CC
cfe_tbl_msg.h, 1547
CFE_TBL_VALIDATION_ERR_EID
cfe_tbl_events.h, 1533
CFE_TBL_VALIDATION_INF_EID
cfe_tbl_events.h, 1533
CFE_TBL_Validate
cFE Manage Table Content APIs, 483
CFE_TBL_ValidateCmd, 835
CommandHeader, 835
Payload, 835
CFE_TBL_ValidateCmd_Payload, 836
ActiveTableFlag, 836
TableName, 836
CFE_TBL_ValidateCmd_Payload_t
cfe_tbl_msg.h, 1553
CFE_TBL_ValidateCmd_t
cfe_tbl_msg.h, 1553
CFE_TBL_WARN_DUPLICATE
cFE Return Code Defines, 336
CFE_TBL_WARN_NOT_CRITICAL
cFE Return Code Defines, 336
CFE_TBL_WARN_PARTIAL_LOAD
cFE Return Code Defines, 336
CFE_TBL_WARN_SHORT_FILE
cFE Return Code Defines, 336
CFE_TBL_WRITE_CFE_HDR_ERR_EID
cfe_tbl_events.h, 1533
CFE_TBL_WRITE_DUMP_INF_EID
cfe_tbl_events.h, 1534

CFE_TBL_WRITE_REG_DUMP_INF_EID
cfe_tbl_events.h, 1534
CFE_TBL_WRITE_TBL_HDR_ERR_EID
cfe_tbl_events.h, 1534
CFE_TBL_WRITE_TBL_IMG_ERR_EID
cfe_tbl_events.h, 1535
CFE_TBL_WRITE_TBL_REG_ERR_EID
cfe_tbl_events.h, 1535
CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID
cfe_tbl_events.h, 1535
CFE_TEST_CMD_MID
cpu1_msgids.h, 1159
CFE_TEST_HK_TLM_MID
cpu1_msgids.h, 1159
CFE_TIME_1HZ_CFG_EID
cfe_time_events.h, 1555
CFE_TIME_1HZ_CMD_MID
cpu1_msgids.h, 1159
CFE_TIME_1HZ_EID
cfe_time_events.h, 1555
CFE_TIME_1HzCmd_t
cfe_time_msg.h, 1587
CFE_TIME_ADD_1HZ_ADJUSTMENT_CC
cfe_time_msg.h, 1570
CFE_TIME_ADD_ADJUST_CC
cfe_time_msg.h, 1571
CFE_TIME_ADD_DELAY_CC
cfe_time_msg.h, 1572
CFE_TIME_Add
cFE Time Arithmetic APIs, 505
CFE_TIME_Add1HZAdjustmentCmd_t
cfe_time_msg.h, 1588
CFE_TIME_AddAdjustCmd_t
cfe_time_msg.h, 1588
CFE_TIME_AddDelayCmd_t
cfe_time_msg.h, 1588
CFE_TIME_AdjustDirection
cfe_time_extern_typedefs.h, 1343
CFE_TIME_AdjustDirection_Enum_t
cfe_time_extern_typedefs.h, 1341
CFE_TIME_BAD_ARGUMENT
cFE Return Code Defines, 337
CFE_TIME_CALLBACK_NOT_REGISTERED
cFE Return Code Defines, 337
CFE_TIME_CC_ERR_EID
cfe_time_events.h, 1555
CFE_TIME_CMD_MID
cpu1_msgids.h, 1159
CFE_TIME_ClockState
cfe_time_extern_typedefs.h, 1344
CFE_TIME_ClockState_Enum_t
cfe_time_extern_typedefs.h, 1341
CFE_TIME_Compare
cFE Time Arithmetic APIs, 506
cfe_time_api_typedefs.h, 1339
CFE_TIME_Compare_t
cfe_time_api_typedefs.h, 1339
CFE_TIME_Copy
cfe_time.h, 1337
CFE_TIME_DATA_CMD_MID
cpu1_msgids.h, 1159
CFE_TIME_DELAY_CFG_EID
cfe_time_events.h, 1556
CFE_TIME_DELAY_EID
cfe_time_events.h, 1556
CFE_TIME_DELAY_ERR_EID
cfe_time_events.h, 1556
CFE_TIME_DELTA_CFG_EID
cfe_time_events.h, 1557
CFE_TIME_DELTA_EID
cfe_time_events.h, 1557
CFE_TIME_DELTA_ERR_EID
cfe_time_events.h, 1557
CFE_TIME_DIAG_EID
cfe_time_events.h, 1558
CFE_TIME_DIAG_TLM_MID
cpu1_msgids.h, 1160
CFE_TIME_DiagnosticTlm, 837
Payload, 837
TelemetryHeader, 837
CFE_TIME_DiagnosticTlm_Payload, 837
AtToneDelay, 840
AtToneLatch, 840
AtToneLeapSeconds, 840
AtToneMET, 840
AtToneSTCF, 840
ClockFlyState, 841
ClockSetState, 841
ClockSignal, 841
ClockSource, 841
ClockStateAPI, 842
ClockStateFlags, 842
CurrentLatch, 842
CurrentMET, 842
CurrentTAI, 843
CurrentUTC, 843
DataStoreStatus, 843
DelayDirection, 843
Forced2Fly, 844
LocalIntCounter, 844
LocalTaskCounter, 844
MaxElapsed, 844
MaxLocalClock, 845
MinElapsed, 845
OneHzAdjust, 845
OneHzDirection, 845
OneTimeAdjust, 846
OneTimeDirection, 846

ServerFlyState, 846
TimeSinceTone, 846
ToneDataCounter, 847
ToneDataLatch, 847
ToneIntCounter, 847
ToneIntErrorCounter, 847
ToneMatchCounter, 848
ToneMatchErrorCounter, 848
ToneOverLimit, 848
ToneSignalCounter, 848
ToneSignalLatch, 849
ToneTaskCounter, 849
ToneUnderLimit, 849
VersionCounter, 849
VirtualMET, 850
CFE_TIME_DiagnosticTlm_Payload_t
cfe_time_msg.h, 1588
CFE_TIME_DiagnosticTlm_t
cfe_time_msg.h, 1588
CFE_TIME_ExternalGPS
cFE External Time Source APIs, 511
CFE_TIME_ExternalMET
cFE External Time Source APIs, 512
CFE_TIME_ExternalTime
cFE External Time Source APIs, 512
CFE_TIME_ExternalTone
cFE External Time Source APIs, 513
CFE_TIME_FLAG_ADD1HZ
cFE Clock State Flag Defines, 520
CFE_TIME_FLAG_ADDADJ
cFE Clock State Flag Defines, 520
CFE_TIME_FLAG_ADDTCL
cFE Clock State Flag Defines, 521
CFE_TIME_FLAG_CLKSET
cFE Clock State Flag Defines, 521
CFE_TIME_FLAG_CMDFLY
cFE Clock State Flag Defines, 521
CFE_TIME_FLAG_FLYING
cFE Clock State Flag Defines, 521
CFE_TIME_FLAG_GDTONE
cFE Clock State Flag Defines, 521
CFE_TIME_FLAG_REFERR
cFE Clock State Flag Defines, 522
CFE_TIME_FLAG_SERVER
cFE Clock State Flag Defines, 522
CFE_TIME_FLAG_SIGPRI
cFE Clock State Flag Defines, 522
CFE_TIME_FLAG_SRCINT
cFE Clock State Flag Defines, 522
CFE_TIME_FLAG_SRVFLY
cFE Clock State Flag Defines, 522
CFE_TIME_FLAG_UNUSED
cFE Clock State Flag Defines, 522
CFE_TIME_FLY_OFF_EID
cfe_time_events.h, 1558
CFE_TIME_FLY_ON_EID
cfe_time_events.h, 1558
CFE_TIME_FakeToneCmd_t
cfe_time_msg.h, 1588
CFE_TIME_FlagBit
cfe_time_extern_typedefs.h, 1344
CFE_TIME_FlagBit_Enum_t
cfe_time_extern_typedefs.h, 1342
CFE_TIME_FlywheelState
cfe_time_extern_typedefs.h, 1345
CFE_TIME_FlywheelState_Enum_t
cfe_time_extern_typedefs.h, 1342
CFE_TIME_GetClockInfo
cFE Get Time Information APIs, 502
CFE_TIME_GetClockState
cFE Get Time Information APIs, 502
CFE_TIME_GetLeapSeconds
cFE Get Time Information APIs, 503
CFE_TIME_GetMETseconds
cFE Get Current Time APIs, 498
CFE_TIME_GetMETsubsecs
cFE Get Current Time APIs, 499
CFE_TIME_GetMET
cFE Get Current Time APIs, 498
CFE_TIME_GetSTCF
cFE Get Time Information APIs, 503
CFE_TIME_GetTAI
cFE Get Current Time APIs, 499
CFE_TIME_GetTime
cFE Get Current Time APIs, 500
CFE_TIME_GetUTC
cFE Get Current Time APIs, 501
CFE_TIME_HK_TLM_MID
cpu1_msgids.h, 1160
CFE_TIME_HousekeepingTlm, 850
Payload, 851
TelemetryHeader, 851
CFE_TIME_HousekeepingTlm_Payload, 851
ClockStateAPI, 852
ClockStateFlags, 852
CommandCounter, 852
CommandErrorCounter, 853
LeapSeconds, 853
Seconds1HzAdj, 853
SecondsDelay, 853
SecondsMET, 854
SecondsSTCF, 854
Subsecs1HzAdj, 854
SubsecsDelay, 854
SubsecsMET, 855
SubsecsSTCF, 855
CFE_TIME_HousekeepingTlm_Payload_t
cfe_time_msg.h, 1589

CFE_TIME_HousekeepingTlm_t
cfe_time_msg.h, 1589
CFE_TIME_ID_ERR_EID
cfe_time_events.h, 1559
CFE_TIME_INIT_EID
cfe_time_events.h, 1559
CFE_TIME_INTERNAL_ONLY
cFE Return Code Defines, 337
CFE_TIME_LEAPS_CFG_EID
cfe_time_events.h, 1559
CFE_TIME_LEAPS_EID
cfe_time_events.h, 1560
CFE_TIME_LEN_ERR_EID
cfe_time_events.h, 1560
CFE_TIME_LeapsCmd_Payload, 855
LeapSeconds, 856
CFE_TIME_LeapsCmd_Payload_t
cfe_time_msg.h, 1589
CFE_TIME_Local1HzISR
cFE Miscellaneous Time APIs, 516
CFE_TIME_MET2SCTime
cFE Time Conversion APIs, 508
CFE_TIME_MET_CFG_EID
cfe_time_events.h, 1560
CFE_TIME_MET_EID
cfe_time_events.h, 1561
CFE_TIME_MET_ERR_EID
cfe_time_events.h, 1561
CFE_TIME_Micro2SubSecs
cFE Time Conversion APIs, 508
CFE_TIME_NOOP_CC
cfe_time_msg.h, 1573
CFE_TIME_NOOP_EID
cfe_time_events.h, 1561
CFE_TIME_NOT_IMPLEMENTED
cFE Return Code Defines, 337
CFE_TIME_NoArgsCmd, 856
CommandHeader, 857
CFE_TIME_NoArgsCmd_t
cfe_time_msg.h, 1589
CFE_TIME_NoopCmd_t
cfe_time_msg.h, 1589
CFE_TIME_OUT_OF_RANGE
cFE Return Code Defines, 338
CFE_TIME_OneHzAdjustmentCmd, 857
CommandHeader, 857
Payload, 857
CFE_TIME_OneHzAdjustmentCmd_Payload, 858
Seconds, 858
Subseconds, 858
CFE_TIME_OneHzAdjustmentCmd_Payload_t
cfe_time_msg.h, 1589
CFE_TIME_OneHzAdjustmentCmd_t
cfe_time_msg.h, 1590
CFE_TIME_PRINTED_STRING_SIZE
cfe_time_api_typedefs.h, 1338
CFE_TIME_Print
cFE Miscellaneous Time APIs, 516
CFE_TIME_RESET_COUNTERS_CC
cfe_time_msg.h, 1574
CFE_TIME_RESET_EID
cfe_time_events.h, 1562
CFE_TIME_RegisterSynchCallback
cFE External Time Source APIs, 514
CFE_TIME_ResetCountersCmd_t
cfe_time_msg.h, 1590
CFE_TIME_SEND_CMD_MID
cpu1_msgids.h, 1160
CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
cfe_time_msg.h, 1575
CFE_TIME_SEND_HK_MID
cpu1_msgids.h, 1160
CFE_TIME_SERVICE
cfe_error.h, 1257
CFE_TIME_SET_LEAP_SECONDS_CC
cfe_time_msg.h, 1576
CFE_TIME_SET_MET_CC
cfe_time_msg.h, 1577
CFE_TIME_SET_SIGNAL_CC
cfe_time_msg.h, 1578
CFE_TIME_SET_SOURCE_CC
cfe_time_msg.h, 1579
CFE_TIME_SET_STATE_CC
cfe_time_msg.h, 1580
CFE_TIME_SET_STCF_CC
cfe_time_msg.h, 1582
CFE_TIME_SET_TIME_CC
cfe_time_msg.h, 1583
CFE_TIME_SIGNAL_CFG_EID
cfe_time_events.h, 1562
CFE_TIME_SIGNAL_EID
cfe_time_events.h, 1562
CFE_TIME_SIGNAL_ERR_EID
cfe_time_events.h, 1563
CFE_TIME_SOURCE_CFG_EID
cfe_time_events.h, 1563
CFE_TIME_SOURCE_EID
cfe_time_events.h, 1563
CFE_TIME_SOURCE_ERR_EID
cfe_time_events.h, 1564
CFE_TIME_STATE_EID
cfe_time_events.h, 1564
CFE_TIME_STATE_ERR_EID
cfe_time_events.h, 1564
CFE_TIME_STCF_CFG_EID
cfe_time_events.h, 1565
CFE_TIME_STCF_EID
cfe_time_events.h, 1565

CFE_TIME_STCF_ERR_EID
cfe_time_events.h, 1565
CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
cfe_time_msg.h, 1584
CFE_TIME_SUB_ADJUST_CC
cfe_time_msg.h, 1585
CFE_TIME_SUB_DELAY_CC
cfe_time_msg.h, 1586
CFE_TIME_SendDiagnosticCmd_t
cfe_time_msg.h, 1590
CFE_TIME_SetLeapSecondsCmd, 859
CommandHeader, 859
Payload, 859
CFE_TIME_SetLeapSecondsCmd_t
cfe_time_msg.h, 1590
CFE_TIME_SetMETCmd_t
cfe_time_msg.h, 1590
CFE_TIME_SetSTCFCmd_t
cfe_time_msg.h, 1591
CFE_TIME_SetSignalCmd, 860
CommandHeader, 860
Payload, 860
CFE_TIME_SetSignalCmd_t
cfe_time_msg.h, 1590
CFE_TIME_SetSourceCmd, 861
CommandHeader, 861
Payload, 861
CFE_TIME_SetSourceCmd_t
cfe_time_msg.h, 1591
CFE_TIME_SetState
cfe_time_extern_typedefs.h, 1345
CFE_TIME_SetState_Enum_t
cfe_time_extern_typedefs.h, 1342
CFE_TIME_SetStateCmd, 862
CommandHeader, 862
Payload, 862
CFE_TIME_SetStateCmd_t
cfe_time_msg.h, 1591
CFE_TIME_SetTimeCmd_t
cfe_time_msg.h, 1591
CFE_TIME_SignalCmd_Payload, 863
ToneSource, 863
CFE_TIME_SignalCmd_Payload_t
cfe_time_msg.h, 1591
CFE_TIME_SourceCmd_Payload, 864
TimeSource, 864
CFE_TIME_SourceCmd_Payload_t
cfe_time_msg.h, 1591
CFE_TIME_SourceSelect
cfe_time_extern_typedefs.h, 1345
CFE_TIME_SourceSelect_Enum_t
cfe_time_extern_typedefs.h, 1343
CFE_TIME_StateCmd_Payload, 864
ClockState, 865
CFE_TIME_StateCmd_Payload_t
cfe_time_msg.h, 1592
CFE_TIME_Sub1HZAdjustmentCmd_t
cfe_time_msg.h, 1592
CFE_TIME_Sub2MicroSecs
cFE Time Conversion APIs, 509
CFE_TIME_SubAdjustCmd_t
cfe_time_msg.h, 1592
CFE_TIME_SubDelayCmd_t
cfe_time_msg.h, 1592
CFE_TIME_Subtract
cFE Time Arithmetic APIs, 507
CFE_TIME_SynchCallbackPtr_t
cfe_time_api_typedefs.h, 1339
CFE_TIME_SysTime, 865
Seconds, 866
Subseconds, 866
CFE_TIME_SysTime_t
cfe_time_extern_typedefs.h, 1343
CFE_TIME_TIME_CFG_EID
cfe_time_events.h, 1566
CFE_TIME_TIME_EID
cfe_time_events.h, 1566
CFE_TIME_TIME_ERR_EID
cfe_time_events.h, 1566
CFE_TIME_TONE_CMD_MID
cpu1_msgids.h, 1160
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS
cFE Return Code Defines, 338
CFE_TIME_TimeCmd, 866
CommandHeader, 867
Payload, 867
CFE_TIME_TimeCmd_Payload, 867
MicroSeconds, 867
Seconds, 868
CFE_TIME_TimeCmd_Payload_t
cfe_time_msg.h, 1592
CFE_TIME_TimeCmd_t
cfe_time_msg.h, 1592
CFE_TIME_ToneDataCmd, 868
CommandHeader, 868
Payload, 869
CFE_TIME_ToneDataCmd_Payload, 869
AtToneLeapSeconds, 870
AtToneMET, 870
AtToneSTCF, 870
AtToneState, 870
CFE_TIME_ToneDataCmd_Payload_t
cfe_time_msg.h, 1593
CFE_TIME_ToneDataCmd_t
cfe_time_msg.h, 1593
CFE_TIME_ToneSignalCmd_t
cfe_time_msg.h, 1593
CFE_TIME_ToneSignalSelect

cfe_time_extern_typedefs.h, 1345
CFE_TIME_ToneSignalSelect_Enum_t
cfe_time_extern_typedefs.h, 1343
CFE_TIME_UnregisterSynchCallback
cFE External Time Source APIs, 514
CFE_TST
cfe_sb.h, 1319
CFE_VERSION_STRING
cfe_version.h, 1348
CFECoreChecksum
CFE_ES_HousekeepingTlm_Payload, 687
CFEMajorVersion
CFE_ES_HousekeepingTlm_Payload, 687
CFEMinorVersion
CFE_ES_HousekeepingTlm_Payload, 687
CFEMissionRevision
CFE_ES_HousekeepingTlm_Payload, 687
CFERevision
CFE_ES_HousekeepingTlm_Payload, 688
CFS File Manager Command Codes, 278
FM_CONCAT_CC, 278
FM_COPY_CC, 280
FM_CREATE_DIR_CC, 281
FM_DECOMPRESS_CC, 282
FM_DELETE_ALL_CC, 284
FM_DELETE_CC, 285
FM_DELETE_DIR_CC, 287
FM_DELETE_INT_CC, 288
FM_GET_DIR_FILE_CC, 288
FM_GET_DIR_PKT_CC, 290
FM_GET_FILE_INFO_CC, 292
FM_GET_FREE_SPACE_CC, 293
FM_GET_OPEN_FILES_CC, 294
FM_MOVE_CC, 294
FM_NOOP_CC, 296
FM_RENAME_CC, 296
FM_RESET_CC, 298
FM_SET_FILE_PERM_CC, 298
FM_SET_TABLE_STATE_CC, 299
CFS File Manager Command Message IDs, 132
FM_CMD_MID, 132
FM_SEND_HK_MID, 132
CFS File Manager Command Structures, 276
CFS File Manager Event IDs, 146
FM_CC_ERR_EID, 158
FM_CHILD_BROKEN_EID_OFFSET, 158
FM_CHILD_DISABLED_EID_OFFSET, 158
FM_CHILD_EXE_ERR_EID, 159
FM_CHILD_INIT_CREATE_ERR_EID, 159
FM_CHILD_INIT_EID, 159
FM_CHILD_INIT_QSEM_ERR_EID, 160
FM_CHILD_INIT_SEM_ERR_EID, 160
FM_CHILD_NUM_OFFSETS, 161
FM_CHILD_Q_FULL_EID_OFFSET, 161
FM_CHILD_TERM_EMPTYQ_ERR_EID, 161
FM_CHILD_TERM_QIDX_ERR_EID, 162
FM_CHILD_TERM_SEM_ERR_EID, 162
FM_CONCAT_CHILD_BASE_EID, 163
FM_CONCAT_CHILD_BROKEN_ERR_EID, 163
FM_CONCAT_CHILD_DISABLED_ERR_EID, 164
FM_CONCAT_CHILD_FULL_ERR_EID, 164
FM_CONCAT_CMD_EID, 165
FM_CONCAT_OPEN_SRC2_ERR_EID, 165
FM_CONCAT_OPEN_TGT_ERR_EID, 166
FM_CONCAT_OSCPY_ERR_EID, 166
FM_CONCAT_OSRD_ERR_EID, 167
FM_CONCAT_OSWR_ERR_EID, 167
FM_CONCAT_PKT_ERR_EID, 168
FM_CONCAT_SRC1_BASE_EID, 168
FM_CONCAT_SRC1_DNE_ERR_EID, 169
FM_CONCAT_SRC1_INVALID_ERR_EID, 169
FM_CONCAT_SRC1_ISDIR_ERR_EID, 170
FM_CONCAT_SRC1_OPEN_ERR_EID, 170
FM_CONCAT_SRC2_BASE_EID, 171
FM_CONCAT_SRC2_DNE_ERR_EID, 171
FM_CONCAT_SRC2_INVALID_ERR_EID, 172
FM_CONCAT_SRC2_ISDIR_ERR_EID, 172
FM_CONCAT_SRC2_OPEN_ERR_EID, 173
FM_CONCAT_TGT_BASE_EID, 173
FM_CONCAT_TGT_EXIST_ERR_EID, 174
FM_CONCAT_TGT_INVALID_ERR_EID, 174
FM_CONCAT_TGT_ISDIR_ERR_EID, 175
FM_COPY_CHILD_BASE_EID, 175
FM_COPY_CHILD_BROKEN_ERR_EID, 176
FM_COPY_CHILD_DISABLED_ERR_EID, 176
FM_COPY_CHILD_FULL_ERR_EID, 177
FM_COPY_CMD_EID, 177
FM_COPY_OS_ERR_EID, 178
FM_COPY_OVR_ERR_EID, 178
FM_COPY_PKT_ERR_EID, 179
FM_COPY_SRC_BASE_EID, 179
FM_COPY_SRC_DNE_ERR_EID, 180
FM_COPY_SRC_INVALID_ERR_EID, 180
FM_COPY_SRC_ISDIR_ERR_EID, 181
FM_COPY_SRC_UNKNOWN_ERR_EID, 181
FM_COPY_TGT_BASE_EID, 182
FM_COPY_TGT_EXIST_ERR_EID, 182
FM_COPY_TGT_INVALID_ERR_EID, 183
FM_COPY_TGT_ISDIR_ERR_EID, 183
FM_COPY_TGT_ISOPEN_ERR_EID, 184
FM_COPY_TGT_UNKNOWN_ERR_EID, 184
FM_CREATE_DIR_CHILD_BASE_EID, 185
FM_CREATE_DIR_CHILD_BROKEN_ERR_EID, 185
FM_CREATE_DIR_CHILD_DISABLED_ERR_EID, 186
FM_CREATE_DIR_CHILD_FULL_ERR_EID, 186
FM_CREATE_DIR_CMD_EID, 187

FM_CREATE_DIR_OS_ERR_EID, 187
FM_CREATE_DIR_PKT_ERR_EID, 188
FM_CREATE_DIR_SRC_BASE_EID, 188
FM_CREATE_DIR_SRC_DNE_ERR_EID, 189
FM_CREATE_DIR_SRC_INVALID_ERR_EID, 189
FM_CREATE_DIR_SRC_ISDIR_ERR_EID, 190
FM_DECOM_CFE_ERR_EID, 190
FM_DECOM_CHILD_BASE_EID, 191
FM_DECOM_CHILD_BROKEN_ERR_EID, 191
FM_DECOM_CHILD_DISABLED_ERR_EID, 192
FM_DECOM_CHILD_FULL_ERR_EID, 192
FM_DECOM_CMD_EID, 193
FM_DECOM_PKT_ERR_EID, 193
FM_DECOM_SRC_BASE_EID, 194
FM_DECOM_SRC_DNE_ERR_EID, 194
FM_DECOM_SRC_INVALID_ERR_EID, 195
FM_DECOM_SRC_ISDIR_ERR_EID, 195
FM_DECOM_SRC_OPEN_ERR_EID, 196
FM_DECOM_TGT_BASE_EID, 196
FM_DECOM_TGT_EXIST_ERR_EID, 197
FM_DECOM_TGT_INVALID_ERR_EID, 197
FM_DECOM_TGT_ISDIR_ERR_EID, 198
FM_DELETE_ALL_CHILD_BASE_EID, 198
FM_DELETE_ALL_CHILD_BROKEN_ERR_EID, 199
FM_DELETE_ALL_CHILD_DISABLED_ERR_EID, 199
FM_DELETE_ALL_CHILD_FULL_ERR_EID, 200
FM_DELETE_ALL_CMD_EID, 200
FM_DELETE_ALL_FILES_ND_WARNING_EID, 201
FM_DELETE_ALL_OS_ERR_EID, 201
FM_DELETE_ALL_PKT_ERR_EID, 202
FM_DELETE_ALL_SKIP_WARNING_EID, 202
FM_DELETE_ALL_SRC_BASE_EID, 203
FM_DELETE_ALL_SRC_DNE_ERR_EID, 203
FM_DELETE_ALL_SRC_FILE_ERR_EID, 204
FM_DELETE_ALL_SRC_INVALID_ERR_EID, 204
FM_DELETE_CHILD_BASE_EID, 205
FM_DELETE_CHILD_BROKEN_ERR_EID, 205
FM_DELETE_CHILD_DISABLED_ERR_EID, 206
FM_DELETE_CHILD_FULL_ERR_EID, 206
FM_DELETE_CMD_EID, 207
FM_DELETE_DIR_CHILD_BASE_EID, 207
FM_DELETE_DIR_CHILD_BROKEN_ERR_EID, 208
FM_DELETE_DIR_CHILD_DISABLED_ERR_EID, 208
FM_DELETE_DIR_CHILD_FULL_ERR_EID, 209
FM_DELETE_DIR_CMD_EID, 209
FM_DELETE_DIR_EMPTY_ERR_EID, 210
FM_DELETE_DIR_PKT_ERR_EID, 210
FM_DELETE_DIR_SRC_BASE_EID, 211
FM_DELETE_DIR_SRC_DNE_ERR_EID, 211
FM_DELETE_DIR_SRC_INVALID_ERR_EID, 212
FM_DELETE_OPENDIR_OS_ERR_EID, 212
FM_DELETE_OS_ERR_EID, 213
FM_DELETE_PKT_ERR_EID, 213
FM_DELETE_RMDIR_OS_ERR_EID, 214
FM_DELETE_SRC_BASE_EID, 214
FM_DELETE_SRC_DNE_ERR_EID, 215
FM_DELETE_SRC_INVALID_ERR_EID, 215
FM_DELETE_SRC_ISDIR_ERR_EID, 216
FM_DELETE_SRC_OPEN_ERR_EID, 216
FM_EXIT_ERR_EID, 217
FM_FILE_INFO_CHILD_BASE_EID, 217
FM_FILE_INFO_CHILD_BROKEN_ERR_EID, 218
FM_FILE_INFO_CHILD_DISABLED_ERR_EID, 218
FM_FILE_INFO_CHILD_FULL_ERR_EID, 219
FM_FNAME_DNE_EID_OFFSET, 219
FM_FNAME_EXIST_EID_OFFSET, 220
FM_FNAME_INVALID_EID_OFFSET, 220
FM_FNAME_ISCLOSED_EID_OFFSET, 220
FM_FNAME_ISDIR_EID_OFFSET, 220
FM_FNAME_ISFILE_EID_OFFSET, 220
FM_FNAME_ISOPEN_EID_OFFSET, 221
FM_FNAME_NUM_OFFSETS, 221
FM_FNAME_UNKNOWN_EID_OFFSET, 221
FM_GET_DIR_FILE_CHILD_BASE_EID, 221
FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID, 221
FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID, 222
FM_GET_DIR_FILE_CHILD_FULL_ERR_EID, 222
FM_GET_DIR_FILE_CMD_EID, 223
FM_GET_DIR_FILE_OSCREAT_ERR_EID, 223
FM_GET_DIR_FILE_OSOPENDIR_ERR_EID, 224
FM_GET_DIR_FILE_PKT_ERR_EID, 224
FM_GET_DIR_FILE_SRC_BASE_EID, 225
FM_GET_DIR_FILE_SRC_DNE_ERR_EID, 225
FM_GET_DIR_FILE_SRC_INVALID_ERR_EID, 226
FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID, 226
FM_GET_DIR_FILE_TGT_BASE_EID, 227
FM_GET_DIR_FILE_TGT_INVALID_ERR_EID, 227
FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID, 228
FM_GET_DIR_FILE_UPSTATS_ERR_EID, 228
FM_GET_DIR_FILE_WARNING_EID, 229
FM_GET_DIR_FILE_WRBLANK_ERR_EID, 229
FM_GET_DIR_FILE_WRENTRY_ERR_EID, 230
FM_GET_DIR_FILE_WRHDR_ERR_EID, 230
FM_GET_DIR_PKT_CHILD_BASE_EID, 231
FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID, 231
FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID, 232
FM_GET_DIR_PKT_CHILD_FULL_ERR_EID, 232
FM_GET_DIR_PKT_CMD_EID, 233
FM_GET_DIR_PKT_OS_ERR_EID, 233
FM_GET_DIR_PKT_PKT_ERR_EID, 234

FM_GET_DIR_PKT_SRC_BASE_EID, 234
FM_GET_DIR_PKT_SRC_DNE_ERR_EID, 235
FM_GET_DIR_PKT_SRC_INVALID_ERR_EID, 235
FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID, 236
FM_GET_DIR_PKT_WARNING_EID, 236
FM_GET_FILE_INFO_CMD_EID, 237
FM_GET_FILE_INFO_OPEN_ERR_EID, 237
FM_GET_FILE_INFO_PKT_ERR_EID, 238
FM_GET_FILE_INFO_READ_WARNING_EID, 238
FM_GET_FILE_INFO_SRC_ERR_EID, 239
FM_GET_FILE_INFO_STATE_WARNING_EID, 239
FM_GET_FILE_INFO_TYPE_WARNING_EID, 240
FM_GET_FREE_SPACE_CMD_EID, 240
FM_GET_FREE_SPACE_PKT_ERR_EID, 241
FM_GET_FREE_SPACE_TBL_ERR_EID, 241
FM_GET_OPEN_FILES_CMD_EID, 242
FM_GET_OPEN_FILES_PKT_ERR_EID, 242
FM_HK_REQ_ERR_EID, 243
FM_MID_ERR_EID, 243
FM_MOVE_CHILD_BASE_EID, 244
FM_MOVE_CHILD_BROKEN_ERR_EID, 244
FM_MOVE_CHILD_DISABLED_ERR_EID, 245
FM_MOVE_CHILD_FULL_ERR_EID, 245
FM_MOVE_CMD_EID, 246
FM_MOVE_OS_ERR_EID, 246
FM_MOVE_OVR_ERR_EID, 247
FM_MOVE_PKT_ERR_EID, 247
FM_MOVE_SRC_BASE_EID, 248
FM_MOVE_SRC_DNE_ERR_EID, 248
FM_MOVE_SRC_INVALID_ERR_EID, 249
FM_MOVE_SRC_ISDIR_ERR_EID, 249
FM_MOVE_TGT_BASE_EID, 250
FM_MOVE_TGT_EXIST_ERR_EID, 250
FM_MOVE_TGT_INVALID_ERR_EID, 251
FM_MOVE_TGT_ISDIR_ERR_EID, 251
FM_MOVE_TGT_ISOPEN_ERR_EID, 252
FM_MOVE_TGT_UNKNOWN_ERR_EID, 252
FM_NOOP_CMD_EID, 253
FM_NOOP_PKT_ERR_EID, 253
FM_OS_SYS_STAT_ERR_EID, 254
FM_RENAME_CHILD_BASE_EID, 254
FM_RENAME_CHILD_BROKEN_ERR_EID, 255
FM_RENAME_CHILD_DISABLED_ERR_EID, 255
FM_RENAME_CHILD_FULL_ERR_EID, 256
FM_RENAME_CMD_EID, 256
FM_RENAME_OS_ERR_EID, 257
FM_RENAME_OVR_ERR_EID, 257
FM_RENAME_PKT_ERR_EID, 258
FM_RENAME_SRC_BASE_EID, 258
FM_RENAME_SRC_DNE_ERR_EID, 259
FM_RENAME_SRC_INVALID_ERR_EID, 259
FM_RENAME_SRC_ISDIR_ERR_EID, 260
FM_RENAME_TGT_BASE_EID, 260
FM_RENAME_TGT_EXIST_ERR_EID, 261
FM_RENAME_TGT_INVALID_ERR_EID, 261
FM_RENAME_TGT_ISDIR_ERR_EID, 262
FM_RENAME_TGT_ISOPEN_ERR_EID, 262
FM_RENAME_TGT_UNKNOWN_ERR_EID, 263
FM_RESET_CMD_EID, 263
FM_RESET_PKT_ERR_EID, 264
FM_SB_RECEIVE_ERR_EID, 264
FM_SB_RECEIVE_NULL_PTR_ERR_EID, 265
FM_SET_PERM_CMD_EID, 265
FM_SET_PERM_ERR_EID, 266
FM_SET_PERM_OS_ERR_EID, 266
FM_SET_TABLE_STATE_ARG_IDX_ERR_EID, 267
FM_SET_TABLE_STATE_ARG_STATE_ERR_EID, 267
FM_SET_TABLE_STATE_CMD_EID, 268
FM_SET_TABLE_STATE_PKT_ERR_EID, 268
FM_SET_TABLE_STATE_TBL_ERR_EID, 269
FM_SET_TABLE_STATE_UNUSED_ERR_EID, 269
FM_STARTUP_CREAT_PIPE_ERR_EID, 270
FM_STARTUP_EID, 270
FM_STARTUP_EVENTS_ERR_EID, 271
FM_STARTUP_SUBSCRIB_GCMD_ERR_EID, 271
FM_STARTUP_SUBSCRIB_HK_ERR_EID, 272
FM_STARTUP_TABLE_INIT_ERR_EID, 272
FM_TABLE_VERIFY_BAD_STATE_ERR_EID, 273
FM_TABLE_VERIFY_EID, 273
FM_TABLE_VERIFY_EMPTY_ERR_EID, 274
FM_TABLE_VERIFY_NULL_PTR_ERR_EID, 274
FM_TABLE_VERIFY_TOOLONG_ERR_EID, 275
CFS File Manager Mission Configuration, 131
FM_APPMAIN_PERF_ID, 131
FM_CHILD_TASK_PERF_ID, 131
CFS File Manager Platform Configuration, 134
FM_APP_NAME, 135
FM_APP_PIPE_DEPTH, 135
FM_APP_PIPE_NAME, 135
FM_CHILD_FILE_BLOCK_SIZE, 136
FM_CHILD_FILE_LOOP_COUNT, 136
FM_CHILD_FILE_SLEEP_MS, 137
FM_CHILD_QUEUE_DEPTH, 137
FM_CHILD_SEM_NAME, 137
FM_CHILD_STAT_SLEEP_FILECOUNT, 138
FM_CHILD_STAT_SLEEP_MS, 138
FM_CHILD_TASK_NAME, 139
FM_CHILD_TASK_PRIORITY, 139
FM_CHILD_TASK_STACK_SIZE, 140
FM_DIR_LIST_FILE_DEFNAME, 140
FM_DIR_LIST_FILE_ENTRIES, 141
FM_DIR_LIST_FILE_SUBTYPE, 141
FM_DIR_LIST_PKT_ENTRIES, 142
FM_MISSION_REV, 142
FM_TABLE_CFE_NAME, 143
FM_TABLE_DEF_DESC, 143
FM_TABLE_DEF_NAME, 144

FM_TABLE_ENTRY_COUNT, 144
FM_TABLE_FILENAME, 144
FM_TABLE_VALIDATION_ERR, 145
CFS File Manager Telemetry, 277
CFS File Manager Telemetry Message IDs, 133
FM_DIR_LIST_TLM_MID, 133
FM_FILE_INFO_TLM_MID, 133
FM_FREE_SPACE_TLM_MID, 133
FM_HK_TLM_MID, 133
FM_OPEN_FILES_TLM_MID, 133
CFS File Manager Version, 301
FM_MAJOR_VERSION, 301
FM_MINOR_VERSION, 301
FM_REVISION, 301
CRC_Computed
FM_FileInfoPkt_t, 889
CRC
FM_FileInfoPkt_t, 889
ccsds_hdr.h
CCSDS_ExtendedHeader_t, 1465
CCSDS_PrimaryHeader_t, 1465
CdsName
CFE_ES_DeleteCDSCmd_Payload, 680
cfe/cmake/sample_defs/cpu1_msgids.h, 1153
cfe/cmake/sample_defs/cpu1_platform_cfg.h, 1161
cfe/cmake/sample_defs/sample_mission_cfg.h, 1218
cfe/cmake/sample_defs/sample_perfids.h, 1239
cfe/docs/src/cfe_api.dox, 1242
cfe/docs/src/cfe_es.dox, 1242
cfe/docs/src/cfe_evs.dox, 1242
cfe/docs/src/cfe_frontpage.dox, 1242
cfe/docs/src/cfe_glossary.dox, 1242
cfe/docs/src/cfe_sb.dox, 1242
cfe/docs/src/cfe_tbl.dox, 1242
cfe/docs/src/cfe_time.dox, 1242
cfe/docs/src/cfe_xref.dox, 1242
cfe/docs/src/cfs_versions.dox, 1243
cfe/modules/core_api/fsw/inc/cfe.h, 1243
cfe/modules/core_api/fsw/inc/cfe_config.h, 1243
cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h, 1247
cfe/modules/core_api/fsw/inc/cfe_endian.h, 1248
cfe/modules/core_api/fsw/inc/cfe_error.h, 1249
cfe/modules/core_api/fsw/inc/cfe_es.h, 1258
cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.↩h, 1262
cfe/modules/core_api/fsw/inc/cfe_es_extern_typedefs.h, 1269
cfe/modules/core_api/fsw/inc/cfe_evs.h, 1281
cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.↩h, 1283
cfe/modules/core_api/fsw/inc/cfe_evs_extern_typedefs.h, 1286
cfe/modules/core_api/fsw/inc/cfe_fs.h, 1291
cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h, 1292
cfe/modules/core_api/fsw/inc/cfe_fs_extern_typedefs.h, 1295
cfe/modules/core_api/fsw/inc/cfe_msg.h, 1298
cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h, 1300
cfe/modules/core_api/fsw/inc/cfe_resourceid.h, 1308
cfe/modules/core_api/fsw/inc/cfe_resourceid_api_↩typedefs.h, 1315
cfe/modules/core_api/fsw/inc/cfe_sb.h, 1317
cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.↩h, 1319
cfe/modules/core_api/fsw/inc/cfe_sb_extern_typedefs.h, 1324
cfe/modules/core_api/fsw/inc/cfe_tbl.h, 1328
cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.↩h, 1329
cfe/modules/core_api/fsw/inc/cfe_tbl_extern_typedefs.h, 1332
cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h, 1334
cfe/modules/core_api/fsw/inc/cfe_time.h, 1336
cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h, 1338
cfe/modules/core_api/fsw/inc/cfe_time_extern_typedefs.↩h, 1340
cfe/modules/core_api/fsw/inc/cfe_version.h, 1346
cfe/modules/es/fsw/inc/cfe_es_events.h, 1349
cfe/modules/es/fsw/inc/cfe_es_msg.h, 1382
cfe/modules/evs/fsw/inc/cfe_evs_events.h, 1418
cfe/modules/evs/fsw/inc/cfe_evs_msg.h, 1432
cfe/modules/msg/fsw/inc/ccsds_hdr.h, 1464
cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_↩basevalues.h, 1465
cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.↩h, 1466
cfe/modules/sb/fsw/inc/cfe_sb_events.h, 1468
cfe/modules/sb/fsw/inc/cfe_sb_msg.h, 1492
cfe/modules/tbl/fsw/inc/cfe_tbl_events.h, 1510
cfe/modules/tbl/fsw/inc/cfe_tbl_msg.h, 1536
cfe/modules/time/fsw/inc/cfe_time_events.h, 1553
cfe/modules/time/fsw/inc/cfe_time_msg.h, 1567
cfe_config.h
CFE_Config_GetIdByName, 1244
CFE_Config_GetName, 1244
CFE_Config_GetObjPointer, 1245
CFE_Config_GetString, 1245
CFE_Config_GetValue, 1246
CFE_Config_IterateAll, 1246
cfe_config_api_typedefs.h
CFE_CONFIGID_UNDEFINED, 1247
CFE_CONFIGID_C, 1247
CFE_Config_Callback_t, 1248
CFE_ConfigId_t, 1248
cfe_endian.h

CFE_MAKE_BIG16, 1249
CFE_MAKE_BIG32, 1249
cfe_error.h
CFE_EVENTS_SERVICE, 1255
CFE_EXECUTIVE_SERVICE, 1255
CFE_FILE_SERVICE, 1256
CFE_GENERIC_SERVICE, 1256
CFE_SERVICE_BITMASK, 1256
CFE_SEVERITY_BITMASK, 1256
CFE_SEVERITY_ERROR, 1256
CFE_SEVERITY_INFO, 1257
CFE_SEVERITY_SUCCESS, 1257
CFE_SOFTWARE_BUS_SERVICE, 1257
CFE_Status_t, 1258
CFE_TABLE_SERVICE, 1257
CFE_TIME_SERVICE, 1257
cfe_es.h
CFE_ES_DBIT, 1261
CFE_ES_DTEST, 1261
CFE_ES_TEST_LONG_MASK, 1262
OS_PRINTF, 1262
cfe_es_api_typedefs.h
CFE_ES_APP_RESTART, 1264
CFE_ES_APPID_UNDEFINED, 1264
CFE_ES_APPID_C, 1264
CFE_ES_CDS_BAD_HANDLE, 1264
CFE_ES_CDSHANDLE_C, 1264
CFE_ES_COUNTERID_UNDEFINED, 1265
CFE_ES_COUNTERID_C, 1265
CFE_ES_ChildTaskMainFuncPtr_t, 1267
CFE_ES_LIBID_UNDEFINED, 1265
CFE_ES_LIBID_C, 1265
CFE_ES_LibraryEntryFuncPtr_t, 1267
CFE_ES_MEMHANDLE_UNDEFINED, 1265
CFE_ES_MEMHANDLE_C, 1265
CFE_ES_MEMPOOLBUF_C, 1266
CFE_ES_MemPoolBuf_t, 1268
CFE_ES_NO_MUTEX, 1266
CFE_ES_PoolAlign_t, 1268
CFE_ES_STATIC_POOL_TYPE, 1266
CFE_ES_StackPointer_t, 1268
CFE_ES_TASK_STACK_ALLOCATE, 1266
CFE_ES_TASKID_UNDEFINED, 1267
CFE_ES_TASKID_C, 1267
CFE_ES_TaskEntryFuncPtr_t, 1268
CFE_ES_USE_MUTEX, 1267
cfe_es_events.h
CFE_ES_ALL_APPS_EID, 1352
CFE_ES_BOOT_ERR_EID, 1352
CFE_ES_BUILD_INF_EID, 1353
CFE_ES_CC1_ERR_EID, 1353
CFE_ES_CDS_DELETE_ERR_EID, 1354
CFE_ES_CDS_DELETE_TBL_ERR_EID, 1354
CFE_ES_CDS_DELETED_INFO_EID, 1354
CFE_ES_CDS_DUMP_ERR_EID, 1355
CFE_ES_CDS_NAME_ERR_EID, 1355
CFE_ES_CDS_OWNER_ACTIVE_EID, 1355
CFE_ES_CDS_REG_DUMP_INF_EID, 1356
CFE_ES_CDS_REGISTER_ERR_EID, 1356
CFE_ES_CREATING_CDS_DUMP_ERR_EID, 1356
CFE_ES_ERLOG1_INF_EID, 1357
CFE_ES_ERLOG2_EID, 1357
CFE_ES_ERLOG2_ERR_EID, 1357
CFE_ES_ERLOG_PENDING_ERR_EID, 1358
CFE_ES_ERR_SYSLOGMODE_EID, 1358
CFE_ES_ERREXIT_APP_ERR_EID, 1358
CFE_ES_ERREXIT_APP_INF_EID, 1359
CFE_ES_EXIT_APP_ERR_EID, 1359
CFE_ES_EXIT_APP_INF_EID, 1360
CFE_ES_FILEWRITE_ERR_EID, 1360
CFE_ES_INIT_INF_EID, 1360
CFE_ES_INITSTATS_INF_EID, 1361
CFE_ES_INVALID_POOL_HANDLE_ERR_EID, 1361
CFE_ES_LEN_ERR_EID, 1361
CFE_ES_MID_ERR_EID, 1362
CFE_ES_NOOP_INF_EID, 1362
CFE_ES_ONE_APP_EID, 1362
CFE_ES_ONE_APPID_ERR_EID, 1363
CFE_ES_ONE_ERR_EID, 1363
CFE_ES_OSCREATE_ERR_EID, 1363
CFE_ES_PCR_ERR1_EID, 1364
CFE_ES_PCR_ERR2_EID, 1364
CFE_ES_PERF_DATAWRITTEN_EID, 1364
CFE_ES_PERF_FILTMSKCMD_EID, 1365
CFE_ES_PERF_FILTMSKERR_EID, 1365
CFE_ES_PERF_LOG_ERR_EID, 1365
CFE_ES_PERF_STARTCMD_EID, 1366
CFE_ES_PERF_STARTCMD_ERR_EID, 1366
CFE_ES_PERF_STARTCMD_TRIG_ERR_EID, 1366
CFE_ES_PERF_STOPCMD_EID, 1367
CFE_ES_PERF_STOPCMD_ERR2_EID, 1367
CFE_ES_PERF_TRIGMSKCMD_EID, 1367
CFE_ES_PERF_TRIGMSKERR_EID, 1368
CFE_ES_RELOAD_APP_DBG_EID, 1368
CFE_ES_RELOAD_APP_ERR1_EID, 1368
CFE_ES_RELOAD_APP_ERR2_EID, 1369
CFE_ES_RELOAD_APP_ERR3_EID, 1369
CFE_ES_RELOAD_APP_ERR4_EID, 1369
CFE_ES_RELOAD_APP_INF_EID, 1370
CFE_ES_RESET_INF_EID, 1370
CFE_ES_RESET_PR_COUNT_EID, 1370
CFE_ES_RESTART_APP_DBG_EID, 1371
CFE_ES_RESTART_APP_ERR1_EID, 1371
CFE_ES_RESTART_APP_ERR2_EID, 1371
CFE_ES_RESTART_APP_ERR3_EID, 1372
CFE_ES_RESTART_APP_ERR4_EID, 1372

CFE_ES_RESTART_APP_INF_EID, 1372
CFE_ES_SET_MAX_PR_COUNT_EID, 1373
CFE_ES_START_ERR_EID, 1373
CFE_ES_START_EXC_ACTION_ERR_EID, 1373
CFE_ES_START_INF_EID, 1374
CFE_ES_START_INVALID_ENTRY_POINT_ERR↩ _EID, 1374
CFE_ES_START_INVALID_FILENAME_ERR_EID, 1374
CFE_ES_START_NULL_APP_NAME_ERR_EID, 1375
CFE_ES_START_PRIORITY_ERR_EID, 1375
CFE_ES_STOP_DBG_EID, 1375
CFE_ES_STOP_ERR1_EID, 1376
CFE_ES_STOP_ERR2_EID, 1376
CFE_ES_STOP_ERR3_EID, 1376
CFE_ES_STOP_INF_EID, 1377
CFE_ES_SYSLOG1_INF_EID, 1377
CFE_ES_SYSLOG2_EID, 1377
CFE_ES_SYSLOG2_ERR_EID, 1378
CFE_ES_SYSLOGMODE_EID, 1378
CFE_ES_TASKINFO_EID, 1378
CFE_ES_TASKINFO_OSCREATE_ERR_EID, 1379
CFE_ES_TASKINFO_WR_ERR_EID, 1379
CFE_ES_TASKINFO_WRHDR_ERR_EID, 1379
CFE_ES_TASKWR_ERR_EID, 1380
CFE_ES_TLM_POOL_STATS_INFO_EID, 1380
CFE_ES_VERSION_INF_EID, 1380
CFE_ES_WRHDR_ERR_EID, 1381
CFE_ES_WRITE_CFE_HDR_ERR_EID, 1381

cfe_es_extern_typedefs.h
CFE_ES_AppId_t, 1272
CFE_ES_AppInfo_t, 1272
CFE_ES_AppState, 1278
CFE_ES_AppState_Enum_t, 1272
CFE_ES_AppType, 1278
CFE_ES_AppType_Enum_t, 1272
CFE_ES_BlockStats_t, 1273
CFE_ES_CDSHandle_t, 1273
CFE_ES_CDSRegDumpRec_t, 1273
CFE_ES_CounterId_t, 1273
CFE_ES_ExceptionAction, 1279
CFE_ES_ExceptionAction_Enum_t, 1274
CFE_ES_LibId_t, 1274
CFE_ES_LogEntryType, 1279
CFE_ES_LogEntryType_Enum_t, 1274
CFE_ES_LogMode, 1279
CFE_ES_LogMode_Enum_t, 1274
CFE_ES_MEMADDRESS_C, 1271
CFE_ES_MEMOFFSET_C, 1271
CFE_ES_MemAddress_t, 1275
CFE_ES_MemHandle_t, 1275
CFE_ES_MemOffset_t, 1275
CFE_ES_MemPoolStats_t, 1276
CFE_ES_RunStatus, 1280
CFE_ES_RunStatus_Enum_t, 1276
CFE_ES_SystemState, 1280
CFE_ES_SystemState_Enum_t, 1276
CFE_ES_TaskId_t, 1277
CFE_ES_TaskInfo_t, 1277
CFE_ES_TaskPriority_Atom_t, 1277

cfe_es_msg.h
CFE_ES_AppNameCmd_Payload_t, 1410
CFE_ES_AppNameCmd_t, 1410
CFE_ES_AppReloadCmd_Payload_t, 1410
CFE_ES_CLEAR_ER_LOG_CC, 1386
CFE_ES_CLEAR_SYSLOG_CC, 1386
CFE_ES_ClearERLogCmd_t, 1410
CFE_ES_ClearSysLogCmd_t, 1410
CFE_ES_DELETE_CDS_CC, 1387
CFE_ES_DUMP_CDS_REGISTRY_CC, 1388
CFE_ES_DeleteCDSCmd_Payload_t, 1410
CFE_ES_DeleteCDSCmd_t, 1410
CFE_ES_DumpCDSRegistryCmd_Payload_t, 1411
CFE_ES_DumpCDSRegistryCmd_t, 1411
CFE_ES_FileNameCmd_Payload_t, 1411
CFE_ES_FileNameCmd_t, 1411
CFE_ES_HousekeepingTlm_Payload_t, 1411
CFE_ES_HousekeepingTlm_t, 1411
CFE_ES_MemStatsTlm_t, 1412
CFE_ES_NOOP_CC, 1389
CFE_ES_NoArgsCmd_t, 1412
CFE_ES_NoopCmd_t, 1412
CFE_ES_OVER_WRITE_SYSLOG_CC, 1390
CFE_ES_OneAppTlm_Payload_t, 1412
CFE_ES_OneAppTlm_t, 1412
CFE_ES_OverWriteSysLogCmd_Payload_t, 1412
CFE_ES_OverWriteSysLogCmd_t, 1413
CFE_ES_PoolStatsTlm_Payload_t, 1413
CFE_ES_QUERY_ALL_CC, 1391
CFE_ES_QUERY_ALL_TASKS_CC, 1392
CFE_ES_QUERY_ONE_CC, 1393
CFE_ES_QueryAllCmd_t, 1413
CFE_ES_QueryAllTasksCmd_t, 1413
CFE_ES_QueryOneCmd_t, 1413
CFE_ES_RELOAD_APP_CC, 1394
CFE_ES_RESET_COUNTERS_CC, 1395
CFE_ES_RESET_PR_COUNT_CC, 1396
CFE_ES_RESTART_APP_CC, 1397
CFE_ES_RESTART_CC, 1398
CFE_ES_ReloadAppCmd_t, 1413
CFE_ES_ResetCountersCmd_t, 1414
CFE_ES_ResetPRCountCmd_t, 1414
CFE_ES_RestartAppCmd_t, 1414
CFE_ES_RestartCmd_Payload_t, 1414
CFE_ES_RestartCmd_t, 1414
CFE_ES_SEND_MEM_POOL_STATS_CC, 1399
CFE_ES_SET_MAX_PR_COUNT_CC, 1400

CFE_ES_SET_PERF_FILTER_MASK_CC, 1401
CFE_ES_SET_PERF_TRIGGER_MASK_CC, 1402
CFE_ES_START_APP_CC, 1403
CFE_ES_START_PERF_DATA_CC, 1404
CFE_ES_STOP_APP_CC, 1405
CFE_ES_STOP_PERF_DATA_CC, 1406
CFE_ES_SendMemPoolStatsCmd_Payload_t, 1414
CFE_ES_SendMemPoolStatsCmd_t, 1415
CFE_ES_SetMaxPRCountCmd_Payload_t, 1415
CFE_ES_SetMaxPRCountCmd_t, 1415
CFE_ES_SetPerfFilterMaskCmd_Payload_t, 1415
CFE_ES_SetPerfFilterMaskCmd_t, 1415
CFE_ES_SetPerfTrigMaskCmd_Payload_t, 1416
CFE_ES_SetPerfTriggerMaskCmd_t, 1415
CFE_ES_StartAppCmd_Payload_t, 1416
CFE_ES_StartAppCmd_t, 1416
CFE_ES_StartPerfCmd_Payload_t, 1416
CFE_ES_StartPerfDataCmd_t, 1416
CFE_ES_StopAppCmd_t, 1416
CFE_ES_StopPerfCmd_Payload_t, 1417
CFE_ES_StopPerfDataCmd_t, 1417
CFE_ES_WRITE_ER_LOG_CC, 1407
CFE_ES_WRITE_SYSLOG_CC, 1408
CFE_ES_WriteERLogCmd_t, 1417
CFE_ES_WriteSysLogCmd_t, 1417
cfe_evs.h
CFE_EVS_Send, 1281
CFE_EVS_SendCrit, 1282
CFE_EVS_SendDbg, 1282
CFE_EVS_SendErr, 1282
CFE_EVS_SendInfo, 1282
cfe_evs_api_typedefs.h
CFE_EVS_BinFilter_t, 1286
CFE_EVS_EVERY_FOURTH_ONE, 1284
CFE_EVS_EVERY_OTHER_ONE, 1284
CFE_EVS_EVERY_OTHER_TWO, 1284
CFE_EVS_FIRST_16_STOP, 1284
CFE_EVS_FIRST_32_STOP, 1284
CFE_EVS_FIRST_4_STOP, 1284
CFE_EVS_FIRST_64_STOP, 1285
CFE_EVS_FIRST_8_STOP, 1285
CFE_EVS_FIRST_ONE_STOP, 1285
CFE_EVS_FIRST_TWO_STOP, 1285
CFE_EVS_NO_FILTER, 1285
cfe_evs_events.h
CFE_EVS_ADDFILTER_EID, 1419
CFE_EVS_DELFILTER_EID, 1419
CFE_EVS_DISAPPENTTYPE_EID, 1420
CFE_EVS_DISAPPEVT_EID, 1420
CFE_EVS_DISEVTTYPE_EID, 1420
CFE_EVS_DISPORT_EID, 1421
CFE_EVS_ENAAPPEVT_EID, 1421
CFE_EVS_ENAAPPEVTTYPE_EID, 1421
CFE_EVS_ENAEVTTYPE_EID, 1422
CFE_EVS_ENAPORT_EID, 1422
CFE_EVS_ERR_APPNOREGS_EID, 1422
CFE_EVS_ERR_CC_EID, 1423
CFE_EVS_ERR_CRDATFILE_EID, 1423
CFE_EVS_ERR_CRLOGFILE_EID, 1423
CFE_EVS_ERR_EVTIDNOREGS_EID, 1424
CFE_EVS_ERR_ILLAPPIDRANGE_EID, 1424
CFE_EVS_ERR_ILLEGALFMTMOD_EID, 1424
CFE_EVS_ERR_INVALID_BITMASK_EID, 1425
CFE_EVS_ERR_LOGMODE_EID, 1425
CFE_EVS_ERR_MAXREGSFILTER_EID, 1425
CFE_EVS_ERR_MSGID_EID, 1426
CFE_EVS_ERR_NOAPPIDFOUND_EID, 1426
CFE_EVS_ERR_UNREGISTERED_EVS_APP, 1426
CFE_EVS_ERR_WRDATFILE_EID, 1427
CFE_EVS_ERR_WRLOGFILE_EID, 1427
CFE_EVS_EVT_FILTERED_EID, 1427
CFE_EVS_FILTER_MAX_EID, 1428
CFE_EVS_LEN_ERR_EID, 1428
CFE_EVS_LOGMODE_EID, 1428
CFE_EVS_NOOP_EID, 1429
CFE_EVS_RSTALLFILTER_EID, 1429
CFE_EVS_RSTCNT_EID, 1429
CFE_EVS_RSTEVTCNT_EID, 1430
CFE_EVS_RSTFILTER_EID, 1430
CFE_EVS_SETEVTFMTMOD_EID, 1430
CFE_EVS_SETFILTERMSK_EID, 1431
CFE_EVS_STARTUP_EID, 1431
CFE_EVS_WRDAT_EID, 1431
CFE_EVS_WRLOG_EID, 1432
cfe_evs_extern_typedefs.h
CFE_EVS_EventFilter, 1288
CFE_EVS_EventFilter_Enum_t, 1287
CFE_EVS_EventOutput, 1288
CFE_EVS_EventOutput_Enum_t, 1287
CFE_EVS_EventType, 1290
CFE_EVS_EventType_Enum_t, 1287
CFE_EVS_LogMode, 1290
CFE_EVS_LogMode_Enum_t, 1287
CFE_EVS_MsgFormat, 1290
CFE_EVS_MsgFormat_Enum_t, 1288
cfe_evs_msg.h
CFE_EVS_ADD_EVENT_FILTER_CC, 1436
CFE_EVS_AddEventFilterCmd_t, 1457
CFE_EVS_AppDataCmd_Payload_t, 1457
CFE_EVS_AppNameBitMaskCmd_Payload_t, 1458
CFE_EVS_AppNameBitMaskCmd_t, 1458
CFE_EVS_AppNameCmd_Payload_t, 1458
CFE_EVS_AppNameCmd_t, 1458
CFE_EVS_AppNameEventIDCmd_Payload_t, 1458
CFE_EVS_AppNameEventIDCmd_t, 1458
CFE_EVS_AppNameEventIDMaskCmd_Payload_t, 1459

CFE_EVS_AppNameEventIDMaskCmd_t, 1459
CFE_EVS_AppTlmData_t, 1459
CFE_EVS_BitMaskCmd_Payload_t, 1459
CFE_EVS_BitMaskCmd_t, 1459
CFE_EVS_CLEAR_LOG_CC, 1436
CFE_EVS_CRITICAL_BIT, 1437
CFE_EVS_ClearLogCmd_t, 1459
CFE_EVS_DEBUG_BIT, 1437
CFE_EVS_DELETE_EVENT_FILTER_CC, 1438
CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, 1438
CFE_EVS_DISABLE_APP_EVENTS_CC, 1439
CFE_EVS_DISABLE_EVENT_TYPE_CC, 1440
CFE_EVS_DISABLE_PORTS_CC, 1441
CFE_EVS_DeleteEventFilterCmd_t, 1460
CFE_EVS_DisableAppEventTypeCmd_t, 1460
CFE_EVS_DisableAppEventsCmd_t, 1460
CFE_EVS_DisableEventTypeCmd_t, 1460
CFE_EVS_DisablePortsCmd_t, 1460
CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, 1442
CFE_EVS_ENABLE_APP_EVENTS_CC, 1443
CFE_EVS_ENABLE_EVENT_TYPE_CC, 1444
CFE_EVS_ENABLE_PORTS_CC, 1445
CFE_EVS_ERROR_BIT, 1446
CFE_EVS_EnableAppEventTypeCmd_t, 1461
CFE_EVS_EnableAppEventsCmd_t, 1460
CFE_EVS_EnableEventTypeCmd_t, 1461
CFE_EVS_EnablePortsCmd_t, 1461
CFE_EVS_HousekeepingTlm_Payload_t, 1461
CFE_EVS_HousekeepingTlm_t, 1461
CFE_EVS_INFORMATION_BIT, 1447
CFE_EVS_LogFileCmd_Payload_t, 1461
CFE_EVS_LongEventTlm_Payload_t, 1462
CFE_EVS_LongEventTlm_t, 1462
CFE_EVS_NOOP_CC, 1447
CFE_EVS_NoArgsCmd_t, 1462
CFE_EVS_NoopCmd_t, 1462
CFE_EVS_PORT1_BIT, 1447
CFE_EVS_PORT2_BIT, 1448
CFE_EVS_PORT3_BIT, 1448
CFE_EVS_PORT4_BIT, 1448
CFE_EVS_PacketID_t, 1462
CFE_EVS_RESET_ALL_FILTERS_CC, 1448
CFE_EVS_RESET_APP_COUNTER_CC, 1449
CFE_EVS_RESET_COUNTERS_CC, 1450
CFE_EVS_RESET_FILTER_CC, 1451
CFE_EVS_ResetAllFiltersCmd_t, 1462
CFE_EVS_ResetAppCounterCmd_t, 1462
CFE_EVS_ResetCountersCmd_t, 1463
CFE_EVS_ResetFilterCmd_t, 1463
CFE_EVS_SET_EVENT_FORMAT_MODE_CC, 1452
CFE_EVS_SET_FILTER_CC, 1453
CFE_EVS_SET_LOG_MODE_CC, 1454
CFE_EVS_SetEventFormatMode_Payload_t, 1463
CFE_EVS_SetEventFormatModeCmd_t, 1463
CFE_EVS_SetFilterCmd_t, 1463
CFE_EVS_SetLogMode_Payload_t, 1463
CFE_EVS_SetLogModeCmd_t, 1464
CFE_EVS_ShortEventTlm_Payload_t, 1464
CFE_EVS_ShortEventTlm_t, 1464
CFE_EVS_WRITE_APP_DATA_FILE_CC, 1455
CFE_EVS_WRITE_LOG_DATA_FILE_CC, 1456
CFE_EVS_WriteAppDataFileCmd_t, 1464
CFE_EVS_WriteLogDataFileCmd_t, 1464
cfe_fs_api_typedefs.h
CFE_FS_FileCategory_t, 1294
CFE_FS_FileWriteEvent_t, 1295
CFE_FS_FileWriteGetData_t, 1293
CFE_FS_FileWriteMetaData_t, 1293
CFE_FS_FileWriteOnEvent_t, 1293
cfe_fs_extern_typedefs.h
CFE_FS_FILE_CONTENT_ID, 1296
CFE_FS_HDR_DESC_MAX_LEN, 1296
CFE_FS_Header_t, 1296
CFE_FS_SubType, 1297
CFE_FS_SubType_Enum_t, 1296
cfe_msg_api_typedefs.h
CFE_MSG_ApId_t, 1302
CFE_MSG_BAD_ARGUMENT, 1302
CFE_MSG_Checksum_t, 1303
CFE_MSG_CommandHeader_t, 1303
CFE_MSG_EDSVersion_t, 1303
CFE_MSG_Endian, 1305
CFE_MSG_Endian_t, 1303
CFE_MSG_FcnCode_t, 1303
CFE_MSG_HeaderVersion_t, 1304
CFE_MSG_Message_t, 1304
CFE_MSG_NOT_IMPLEMENTED, 1302
CFE_MSG_PlaybackFlag, 1306
CFE_MSG_PlaybackFlag_t, 1304
CFE_MSG_SegmentationFlag, 1306
CFE_MSG_SegmentationFlag_t, 1304
CFE_MSG_SequenceCount_t, 1304
CFE_MSG_Size_t, 1305
CFE_MSG_Subsystem_t, 1305
CFE_MSG_System_t, 1305
CFE_MSG_TelemetryHeader_t, 1305
CFE_MSG_Type, 1306
CFE_MSG_Type_t, 1305
CFE_MSG_WRONG_MSG_TYPE, 1302
cfe_psp.h
BUFF_SIZE, 1649
CFE_PSP_AttachExceptions, 1658
CFE_PSP_Decompress, 1658
CFE_PSP_ERROR_ADDRESS_MISALIGNED, 1649
CFE_PSP_ERROR_NOT_IMPLEMENTED, 1649

CFE_PSP_ERROR_TIMEOUT, 1649
CFE_PSP_ERROR, 1649
CFE_PSP_EepromPowerDown, 1658
CFE_PSP_EepromPowerUp, 1658
CFE_PSP_EepromWrite16, 1658
CFE_PSP_EepromWrite32, 1659
CFE_PSP_EepromWrite8, 1659
CFE_PSP_EepromWriteDisable, 1659
CFE_PSP_EepromWriteEnable, 1659
CFE_PSP_Exception_CopyContext, 1659
CFE_PSP_Exception_GetCount, 1659
CFE_PSP_Exception_GetSummary, 1660
CFE_PSP_FlushCaches, 1660
CFE_PSP_Get_Dec, 1660
CFE_PSP_Get_Timebase, 1660
CFE_PSP_Get_Timer_Tick, 1661
CFE_PSP_GetBuildNumber, 1661
CFE_PSP_GetCDSSize, 1661
CFE_PSP_GetCFETextSegmentInfo, 1661
CFE_PSP_GetKernelTextSegmentInfo, 1661
CFE_PSP_GetProcessorId, 1662
CFE_PSP_GetProcessorName, 1662
CFE_PSP_GetResetArea, 1662
CFE_PSP_GetRestartType, 1662
CFE_PSP_GetSpacecraftId, 1662
CFE_PSP_GetTime, 1662
CFE_PSP_GetTimerLow32Rollover, 1663
CFE_PSP_GetTimerTicksPerSecond, 1663
CFE_PSP_GetUserReservedArea, 1663
CFE_PSP_GetVersionCodeName, 1663
CFE_PSP_GetVersionNumber, 1664
CFE_PSP_GetVersionString, 1664
CFE_PSP_GetVolatileDiskMem, 1664
CFE_PSP_INVALID_INT_NUM, 1650
CFE_PSP_INVALID_MEM_ADDR, 1650
CFE_PSP_INVALID_MEM_ATTR, 1650
CFE_PSP_INVALID_MEM_RANGE, 1650
CFE_PSP_INVALID_MEM_SIZE, 1650
CFE_PSP_INVALID_MEM_TYPE, 1650
CFE_PSP_INVALID_MEM_WORDSIZE, 1651
CFE_PSP_INVALID_MODULE_ID, 1651
CFE_PSP_INVALID_MODULE_NAME, 1651
CFE_PSP_INVALID_POINTER, 1651
CFE_PSP_InitSSR, 1665
CFE_PSP_MEM_ANY, 1651
CFE_PSP_MEM_ATTR_READWRITE, 1652
CFE_PSP_MEM_ATTR_READ, 1651
CFE_PSP_MEM_ATTR_WRITE, 1652
CFE_PSP_MEM_EEPROM, 1652
CFE_PSP_MEM_INVALID, 1652
CFE_PSP_MEM_RAM, 1652
CFE_PSP_MEM_SIZE_BYTE, 1652
CFE_PSP_MEM_SIZE_DWORD, 1653
CFE_PSP_MEM_SIZE_WORD, 1653
CFE_PSP_Main, 1665
CFE_PSP_MemCpy, 1665
CFE_PSP_MemRangeGet, 1665
CFE_PSP_MemRangeSet, 1665
CFE_PSP_MemRanges, 1665
CFE_PSP_MemRead16, 1666
CFE_PSP_MemRead32, 1666
CFE_PSP_MemRead8, 1666
CFE_PSP_MemSet, 1666
CFE_PSP_MemValidateRange, 1666
CFE_PSP_MemWrite16, 1667
CFE_PSP_MemWrite32, 1667
CFE_PSP_MemWrite8, 1667
CFE_PSP_NO_EXCEPTION_DATA, 1653
CFE_PSP_PANIC_CORE_APP, 1653
CFE_PSP_PANIC_GENERAL_FAILURE, 1653
CFE_PSP_PANIC_MEMORY_ALLOC, 1653
CFE_PSP_PANIC_NONVOL_DISK, 1654
CFE_PSP_PANIC_STARTUP_SEM, 1654
CFE_PSP_PANIC_STARTUP, 1654
CFE_PSP_PANIC_VOLATILE_DISK, 1654
CFE_PSP_Panic, 1667
CFE_PSP_PortRead16, 1667
CFE_PSP_PortRead32, 1667
CFE_PSP_PortRead8, 1668
CFE_PSP_PortWrite16, 1668
CFE_PSP_PortWrite32, 1668
CFE_PSP_PortWrite8, 1668
CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET, 1654
CFE_PSP_RST_SUBTYPE_EXCEPTION, 1654
CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND, 1655
CFE_PSP_RST_SUBTYPE_HW_WATCHDOG, 1655
CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET, 1655
CFE_PSP_RST_SUBTYPE_MAX, 1655
CFE_PSP_RST_SUBTYPE_POWER_CYCLE, 1655
CFE_PSP_RST_SUBTYPE_PUSH_BUTTON, 1656
CFE_PSP_RST_SUBTYPE_RESET_COMMAND, 1656
CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET, 1656
CFE_PSP_RST_TYPE_MAX, 1656
CFE_PSP_RST_TYPE_POWERON, 1656
CFE_PSP_RST_TYPE_PROCESSOR, 1657
CFE_PSP_ReadFromCDS, 1668
CFE_PSP_Restart, 1668
CFE_PSP_SOFT_TIMEBASE_NAME, 1657
CFE_PSP_SUCCESS, 1657
CFE_PSP_SetDefaultExceptionEnvironment, 1669
CFE_PSP_WatchdogDisable, 1669
CFE_PSP_WatchdogEnable, 1669

CFE_PSP_WatchdogGet, 1669
CFE_PSP_WatchdogInit, 1669
CFE_PSP_WatchdogService, 1669
CFE_PSP_WatchdogSet, 1669
CFE_PSP_WriteToCDS, 1669
SIZE_BYTE, 1657
SIZE_HALF, 1657
SIZE_WORD, 1658
cfe_resourceid.h
CFE_RESOURCEID_TEST_DEFINED, 1309
CFE_RESOURCEID_TEST_EQUAL, 1309
CFE_RESOURCEID_TO_ULONG, 1309
CFE_ResourceId_Equal, 1310
CFE_ResourceId_FindNext, 1310
CFE_ResourceId_FromInteger, 1311
CFE_ResourceId_GetBase, 1312
CFE_ResourceId_GetSerial, 1312
CFE_ResourceId_IsDefined, 1313
CFE_ResourceId_ToIndex, 1314
CFE_ResourceId_ToInteger, 1315
cfe_resourceid_api_typedefs.h
CFE_RESOURCEID_RESERVED, 1316
CFE_RESOURCEID_UNDEFINED, 1316
cfe_resourceid_basevalue.h
CFE_RESOURCEID_MAKE_BASE, 1467
CFE_RESOURCEID_MAX, 1467
CFE_RESOURCEID_SHIFT, 1467
cfe_sb.h
CFE_BIT, 1318
CFE_CLR, 1319
CFE_SET, 1319
CFE_TST, 1319
cfe_sb_api_typedefs.h
CFE_SB_Buffer_t, 1324
CFE_SB_DEFAULT_QOS, 1321
CFE_SB_INVALID_MSG_ID, 1321
CFE_SB_INVALID_PIPE, 1321
CFE_SB_MSGID_RESERVED, 1322
CFE_SB_MSGID_UNWRAP_VALUE, 1322
CFE_SB_MSGID_WRAP_VALUE, 1322
CFE_SB_MSGID_C, 1321
CFE_SB_PEND_FOREVER, 1323
CFE_SB_PIPEID_C, 1323
CFE_SB_POLL, 1323
CFE_SB_SUBSCRIPTION, 1324
CFE_SB_UNSUBSCRIPTION, 1324
cfe_sb_events.h
CFE_SB_BAD_CMD_CODE_EID, 1470
CFE_SB_BAD_MSGID_EID, 1470
CFE_SB_BAD_PIPEID_EID, 1471
CFE_SB_CMD0_RCVD_EID, 1471
CFE_SB_CMD1_RCVD_EID, 1471
CFE_SB_CR_PIPE_BAD_ARG_EID, 1472
CFE_SB_CR_PIPE_ERR_EID, 1472
CFE_SB_CR_PIPE_NAME_TAKEN_EID, 1472
CFE_SB_CR_PIPE_NO_FREE_EID, 1473
CFE_SB_DEL_PIPE_ERR1_EID, 1473
CFE_SB_DEL_PIPE_ERR2_EID, 1473
CFE_SB_DEST_BLK_ERR_EID, 1474
CFE_SB_DSBL_RTE1_EID, 1474
CFE_SB_DSBL_RTE2_EID, 1474
CFE_SB_DSBL_RTE3_EID, 1475
CFE_SB_DUP_SUBSCRIP_EID, 1475
CFE_SB_ENBL_RTE1_EID, 1475
CFE_SB_ENBL_RTE2_EID, 1476
CFE_SB_ENBL_RTE3_EID, 1476
CFE_SB_FILEWRITE_ERR_EID, 1476
CFE_SB_FULL_SUB_PKT_EID, 1477
CFE_SB_GET_BUF_ERR_EID, 1477
CFE_SB_GETPIPEIDBYNAME_EID, 1477
CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID, 1478
CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID, 1478
CFE_SB_GETPIPENAME_EID, 1478
CFE_SB_GETPIPENAME_ID_ERR_EID, 1479
CFE_SB_GETPIPENAME_NULL_PTR_EID, 1479
CFE_SB_GETPIPEOPTS_EID, 1479
CFE_SB_GETPIPEOPTS_ID_ERR_EID, 1480
CFE_SB_GETPIPEOPTS_PTR_ERR_EID, 1480
CFE_SB_HASHCOLLISION_EID, 1480
CFE_SB_INIT_EID, 1481
CFE_SB_LEN_ERR_EID, 1481
CFE_SB_MAX_DESTS_MET_EID, 1481
CFE_SB_MAX_MSGS_MET_EID, 1482
CFE_SB_MAX_PIPES_MET_EID, 1482
CFE_SB_MSG_TOO_BIG_EID, 1482
CFE_SB_MSGID_LIM_ERR_EID, 1483
CFE_SB_PART_SUB_PKT_EID, 1483
CFE_SB_PIPE_ADDED_EID, 1483
CFE_SB_PIPE_DELETED_EID, 1484
CFE_SB_Q_FULL_ERR_EID, 1484
CFE_SB_Q_RD_ERR_EID, 1484
CFE_SB_Q_WR_ERR_EID, 1485
CFE_SB_RCV_BAD_ARG_EID, 1485
CFE_SB_SEND_BAD_ARG_EID, 1485
CFE_SB_SEND_INV_MSGID_EID, 1486
CFE_SB_SEND_NO_SUBS_EID, 1486
CFE_SB_SETPIPEOPTS_EID, 1486
CFE_SB_SETPIPEOPTS_ID_ERR_EID, 1487
CFE_SB_SETPIPEOPTS_OWNER_ERR_EID, 1487
CFE_SB_SND_RTG_EID, 1487
CFE_SB_SND_RTG_ERR1_EID, 1488
CFE_SB_SND_STATS_EID, 1488
CFE_SB_SUB_ARG_ERR_EID, 1488
CFE_SB_SUB_INV_CALLER_EID, 1489
CFE_SB_SUB_INV_PIPE_EID, 1489
CFE_SB_SUBSCRIPTION_RCVD_EID, 1489

CFE_SB_SUBSCRIPTION_REMOVED_EID, 1490
CFE_SB_SUBSCRIPTION_RPT_EID, 1490
CFE_SB_UNSUB_ARG_ERR_EID, 1490
CFE_SB_UNSUB_INV_CALLER_EID, 1491
CFE_SB_UNSUB_INV_PIPE_EID, 1491
CFE_SB_UNSUB_NO_SUBS_EID, 1491
cfe_sb_extern_typedefs.h
CFE_SB_MsgId_Atom_t, 1326
CFE_SB_PipeId_t, 1326
CFE_SB_QosPriority, 1327
CFE_SB_QosPriority_Enum_t, 1326
CFE_SB_QosReliability, 1327
CFE_SB_QosReliability_Enum_t, 1326
CFE_SB_RouteId_Atom_t, 1327
CFE_SB_SUB_ENTRIES_PER_PKT, 1325
cfe_sb_msg.h
CFE_SB_AllSubscriptionsTlm_Payload_t, 1505
CFE_SB_AllSubscriptionsTlm_t, 1505
CFE_SB_DISABLE_ROUTE_CC, 1494
CFE_SB_DISABLE_SUB_REPORTING_CC, 1494
CFE_SB_DisableRouteCmd_t, 1505
CFE_SB_DisableSubReportingCmd_t, 1505
CFE_SB_ENABLE_ROUTE_CC, 1495
CFE_SB_ENABLE_SUB_REPORTING_CC, 1496
CFE_SB_EnableRouteCmd_t, 1505
CFE_SB_EnableSubReportingCmd_t, 1505
CFE_SB_HousekeepingTlm_Payload_t, 1506
CFE_SB_HousekeepingTlm_t, 1506
CFE_SB_MsgMapFileEntry_t, 1506
CFE_SB_NOOP_CC, 1497
CFE_SB_NoopCmd_t, 1506
CFE_SB_PipeDepthStats_t, 1506
CFE_SB_PipeInfoEntry_t, 1506
CFE_SB_RESET_COUNTERS_CC, 1498
CFE_SB_ResetCountersCmd_t, 1507
CFE_SB_RouteCmd_Payload_t, 1507
CFE_SB_RouteCmd_t, 1507
CFE_SB_RoutingFileEntry_t, 1507
CFE_SB_SEND_PREV_SUBS_CC, 1499
CFE_SB_SEND_SB_STATS_CC, 1500
CFE_SB_SendPrevSubsCmd_t, 1507
CFE_SB_SendSbStatsCmd_t, 1508
CFE_SB_SingleSubscriptionTlm_Payload_t, 1508
CFE_SB_SingleSubscriptionTlm_t, 1508
CFE_SB_StatsTlm_Payload_t, 1508
CFE_SB_StatsTlm_t, 1508
CFE_SB_SubEntries_t, 1509
CFE_SB_WRITE_MAP_INFO_CC, 1501
CFE_SB_WRITE_PIPE_INFO_CC, 1502
CFE_SB_WRITE_ROUTING_INFO_CC, 1503
CFE_SB_WriteFileInfoCmd_Payload_t, 1509
CFE_SB_WriteFileInfoCmd_t, 1509
CFE_SB_WriteMapInfoCmd_t, 1509
CFE_SB_WritePipeInfoCmd_t, 1509
CFE_SB_WriteRoutingInfoCmd_t, 1509
cfe_tbl_api_typedefs.h
CFE_TBL_BAD_TABLE_HANDLE, 1331
CFE_TBL_CallbackFuncPtr_t, 1331
CFE_TBL_Handle_t, 1331
CFE_TBL_Info_t, 1331
CFE_TBL_MAX_FULL_NAME_LEN, 1331
CFE_TBL_SrcEnum, 1332
CFE_TBL_SrcEnum_t, 1332
cfe_tbl_events.h
CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID, 1512
CFE_TBL_ACTIVATE_ERR_EID, 1513
CFE_TBL_ASSUMED_VALID_INF_EID, 1513
CFE_TBL_CC1_ERR_EID, 1513
CFE_TBL_CDS_DELETE_ERR_EID, 1514
CFE_TBL_CDS_DELETED_INFO_EID, 1514
CFE_TBL_CDS_NOT_FOUND_ERR_EID, 1514
CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID, 1515
CFE_TBL_CREATING_DUMP_FILE_ERR_EID, 1515
CFE_TBL_DUMP_PENDING_ERR_EID, 1515
CFE_TBL_FAIL_HK_SEND_ERR_EID, 1516
CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID, 1516
CFE_TBL_FILE_ACCESS_ERR_EID, 1516
CFE_TBL_FILE_INCOMPLETE_ERR_EID, 1517
CFE_TBL_FILE_LOADED_INF_EID, 1517
CFE_TBL_FILE_STD_HDR_ERR_EID, 1517
CFE_TBL_FILE_SUBTYPE_ERR_EID, 1518
CFE_TBL_FILE_TBL_HDR_ERR_EID, 1518
CFE_TBL_FILE_TOO_BIG_ERR_EID, 1518
CFE_TBL_FILE_TYPE_ERR_EID, 1519
CFE_TBL_HANDLE_ACCESS_ERR_EID, 1519
CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID, 1519
CFE_TBL_IN_REGISTRY_ERR_EID, 1520
CFE_TBL_INIT_INF_EID, 1520
CFE_TBL_INTERNAL_ERROR_ERR_EID, 1520
CFE_TBL_LEN_ERR_EID, 1521
CFE_TBL_LOAD_ABORT_ERR_EID, 1521
CFE_TBL_LOAD_ABORT_INF_EID, 1521
CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID, 1522
CFE_TBL_LOAD_FILENAME_LONG_ERR_EID, 1522
CFE_TBL_LOAD_IN_PROGRESS_ERR_EID, 1522
CFE_TBL_LOAD_PEND_REQ_INF_EID, 1523
CFE_TBL_LOAD_SUCCESS_INF_EID, 1523
CFE_TBL_LOAD_TBLNAME_MISMATCH_ERR_↩ EID, 1523
CFE_TBL_LOAD_TYPE_ERR_EID, 1524
CFE_TBL_LOAD_VAL_ERR_EID, 1524
CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID, 1524
CFE_TBL_LOADING_PENDING_ERR_EID, 1525

CFE_TBL_MID_ERR_EID, 1525
CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID, 1525
CFE_TBL_NO_SUCH_TABLE_ERR_EID, 1526
CFE_TBL_NO_WORK_BUFFERS_ERR_EID, 1526
CFE_TBL_NOOP_INF_EID, 1526
CFE_TBL_NOT_CRITICAL_TBL_ERR_EID, 1527
CFE_TBL_NOT_IN_CRIT_REG_ERR_EID, 1527
CFE_TBL_OVERWRITE_DUMP_INF_EID, 1527
CFE_TBL_OVERWRITE_REG_DUMP_INF_EID, 1528
CFE_TBL_PARTIAL_LOAD_ERR_EID, 1528
CFE_TBL_PROCESSOR_ID_ERR_EID, 1528
CFE_TBL_REGISTER_ERR_EID, 1529
CFE_TBL_RESET_INF_EID, 1529
CFE_TBL_SHARE_ERR_EID, 1529
CFE_TBL_SPACECRAFT_ID_ERR_EID, 1530
CFE_TBL_TLM_REG_CMD_INF_EID, 1530
CFE_TBL_TOO_MANY_DUMPS_ERR_EID, 1530
CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID, 1531
CFE_TBL_UNREGISTER_ERR_EID, 1531
CFE_TBL_UNVALIDATED_ERR_EID, 1531
CFE_TBL_UPDATE_ERR_EID, 1532
CFE_TBL_UPDATE_SUCCESS_INF_EID, 1532
CFE_TBL_VAL_REQ_MADE_INF_EID, 1532
CFE_TBL_VALIDATION_ERR_EID, 1533
CFE_TBL_VALIDATION_INF_EID, 1533
CFE_TBL_WRITE_CFE_HDR_ERR_EID, 1533
CFE_TBL_WRITE_DUMP_INF_EID, 1534
CFE_TBL_WRITE_REG_DUMP_INF_EID, 1534
CFE_TBL_WRITE_TBL_HDR_ERR_EID, 1534
CFE_TBL_WRITE_TBL_IMG_ERR_EID, 1535
CFE_TBL_WRITE_TBL_REG_ERR_EID, 1535
CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID, 1535
cfe_tbl_extern_typedefs.h
CFE_TBL_BufferSelect, 1333
CFE_TBL_BufferSelect_Enum_t, 1333
CFE_TBL_File_Hdr_t, 1333
cfe_tbl_filedef.h
CFE_TBL_FILEDEF, 1335
CFE_TBL_FileDef_t, 1335
cfe_tbl_msg.h
CFE_TBL_ABORT_LOAD_CC, 1538
CFE_TBL_ACTIVATE_CC, 1539
CFE_TBL_AbortLoadCmd_Payload_t, 1549
CFE_TBL_AbortLoadCmd_t, 1549
CFE_TBL_ActivateCmd_Payload_t, 1549
CFE_TBL_ActivateCmd_t, 1549
CFE_TBL_DELETE_CDS_CC, 1540
CFE_TBL_DUMP_CC, 1541
CFE_TBL_DUMP_REGISTRY_CC, 1542
CFE_TBL_DelCDSCmd_Payload_t, 1549
CFE_TBL_DeleteCDSCmd_t, 1549
CFE_TBL_DumpCmd_Payload_t, 1550
CFE_TBL_DumpCmd_t, 1550
CFE_TBL_DumpRegistryCmd_Payload_t, 1550
CFE_TBL_DumpRegistryCmd_t, 1550
CFE_TBL_HousekeepingTlm_Payload_t, 1550
CFE_TBL_HousekeepingTlm_t, 1550
CFE_TBL_LOAD_CC, 1543
CFE_TBL_LoadCmd_Payload_t, 1551
CFE_TBL_LoadCmd_t, 1551
CFE_TBL_NOOP_CC, 1544
CFE_TBL_NoArgsCmd_t, 1551
CFE_TBL_NoopCmd_t, 1551
CFE_TBL_NotifyCmd_Payload_t, 1551
CFE_TBL_NotifyCmd_t, 1552
CFE_TBL_RESET_COUNTERS_CC, 1545
CFE_TBL_ResetCountersCmd_t, 1552
CFE_TBL_SEND_REGISTRY_CC, 1546
CFE_TBL_SendRegistryCmd_Payload_t, 1552
CFE_TBL_SendRegistryCmd_t, 1552
CFE_TBL_TableRegistryTlm_t, 1552
CFE_TBL_TblRegPacket_Payload_t, 1552
CFE_TBL_VALIDATE_CC, 1547
CFE_TBL_ValidateCmd_Payload_t, 1553
CFE_TBL_ValidateCmd_t, 1553
cfe_time.h
CFE_TIME_Copy, 1337
cfe_time_api_typedefs.h
CFE_TIME_Compare, 1339
CFE_TIME_Compare_t, 1339
CFE_TIME_PRINTED_STRING_SIZE, 1338
CFE_TIME_SynchCallbackPtr_t, 1339
cfe_time_events.h
CFE_TIME_1HZ_CFG_EID, 1555
CFE_TIME_1HZ_EID, 1555
CFE_TIME_CC_ERR_EID, 1555
CFE_TIME_DELAY_CFG_EID, 1556
CFE_TIME_DELAY_EID, 1556
CFE_TIME_DELAY_ERR_EID, 1556
CFE_TIME_DELTA_CFG_EID, 1557
CFE_TIME_DELTA_EID, 1557
CFE_TIME_DELTA_ERR_EID, 1557
CFE_TIME_DIAG_EID, 1558
CFE_TIME_FLY_OFF_EID, 1558
CFE_TIME_FLY_ON_EID, 1558
CFE_TIME_ID_ERR_EID, 1559
CFE_TIME_INIT_EID, 1559
CFE_TIME_LEAPS_CFG_EID, 1559
CFE_TIME_LEAPS_EID, 1560
CFE_TIME_LEN_ERR_EID, 1560
CFE_TIME_MET_CFG_EID, 1560
CFE_TIME_MET_EID, 1561
CFE_TIME_MET_ERR_EID, 1561
CFE_TIME_NOOP_EID, 1561
CFE_TIME_RESET_EID, 1562

CFE_TIME_SIGNAL_CFG_EID, 1562
CFE_TIME_SIGNAL_EID, 1562
CFE_TIME_SIGNAL_ERR_EID, 1563
CFE_TIME_SOURCE_CFG_EID, 1563
CFE_TIME_SOURCE_EID, 1563
CFE_TIME_SOURCE_ERR_EID, 1564
CFE_TIME_STATE_EID, 1564
CFE_TIME_STATE_ERR_EID, 1564
CFE_TIME_STCF_CFG_EID, 1565
CFE_TIME_STCF_EID, 1565
CFE_TIME_STCF_ERR_EID, 1565
CFE_TIME_TIME_CFG_EID, 1566
CFE_TIME_TIME_EID, 1566
CFE_TIME_TIME_ERR_EID, 1566
cfe_time_extern_typedefs.h
CFE_TIME_AdjustDirection, 1343
CFE_TIME_AdjustDirection_Enum_t, 1341
CFE_TIME_ClockState, 1344
CFE_TIME_ClockState_Enum_t, 1341
CFE_TIME_FlagBit, 1344
CFE_TIME_FlagBit_Enum_t, 1342
CFE_TIME_FlywheelState, 1345
CFE_TIME_FlywheelState_Enum_t, 1342
CFE_TIME_SetState, 1345
CFE_TIME_SetState_Enum_t, 1342
CFE_TIME_SourceSelect, 1345
CFE_TIME_SourceSelect_Enum_t, 1343
CFE_TIME_SysTime_t, 1343
CFE_TIME_ToneSignalSelect, 1345
CFE_TIME_ToneSignalSelect_Enum_t, 1343
cfe_time_msg.h
CFE_TIME_1HzCmd_t, 1587
CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, 1570
CFE_TIME_ADD_ADJUST_CC, 1571
CFE_TIME_ADD_DELAY_CC, 1572
CFE_TIME_Add1HZAdjustmentCmd_t, 1588
CFE_TIME_AddAdjustCmd_t, 1588
CFE_TIME_AddDelayCmd_t, 1588
CFE_TIME_DiagnosticTlm_Payload_t, 1588
CFE_TIME_DiagnosticTlm_t, 1588
CFE_TIME_FakeToneCmd_t, 1588
CFE_TIME_HousekeepingTlm_Payload_t, 1589
CFE_TIME_HousekeepingTlm_t, 1589
CFE_TIME_LeapsCmd_Payload_t, 1589
CFE_TIME_NOOP_CC, 1573
CFE_TIME_NoArgsCmd_t, 1589
CFE_TIME_NoopCmd_t, 1589
CFE_TIME_OneHzAdjustmentCmd_Payload_t, 1589
CFE_TIME_OneHzAdjustmentCmd_t, 1590
CFE_TIME_RESET_COUNTERS_CC, 1574
CFE_TIME_ResetCountersCmd_t, 1590
CFE_TIME_SEND_DIAGNOSTIC_TLM_CC, 1575
CFE_TIME_SET_LEAP_SECONDS_CC, 1576
CFE_TIME_SET_MET_CC, 1577
CFE_TIME_SET_SIGNAL_CC, 1578
CFE_TIME_SET_SOURCE_CC, 1579
CFE_TIME_SET_STATE_CC, 1580
CFE_TIME_SET_STCF_CC, 1582
CFE_TIME_SET_TIME_CC, 1583
CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, 1584
CFE_TIME_SUB_ADJUST_CC, 1585
CFE_TIME_SUB_DELAY_CC, 1586
CFE_TIME_SendDiagnosticCmd_t, 1590
CFE_TIME_SetLeapSecondsCmd_t, 1590
CFE_TIME_SetMETCmd_t, 1590
CFE_TIME_SetSTCFCmd_t, 1591
CFE_TIME_SetSignalCmd_t, 1590
CFE_TIME_SetSourceCmd_t, 1591
CFE_TIME_SetStateCmd_t, 1591
CFE_TIME_SetTimeCmd_t, 1591
CFE_TIME_SignalCmd_Payload_t, 1591
CFE_TIME_SourceCmd_Payload_t, 1591
CFE_TIME_StateCmd_Payload_t, 1592
CFE_TIME_Sub1HZAdjustmentCmd_t, 1592
CFE_TIME_SubAdjustCmd_t, 1592
CFE_TIME_SubDelayCmd_t, 1592
CFE_TIME_TimeCmd_Payload_t, 1592
CFE_TIME_TimeCmd_t, 1592
CFE_TIME_ToneDataCmd_Payload_t, 1593
CFE_TIME_ToneDataCmd_t, 1593
CFE_TIME_ToneSignalCmd_t, 1593
cfe_version.h
CFE_BUILD_BASELINE, 1347
CFE_BUILD_NUMBER, 1347
CFE_MAJOR_VERSION, 1347
CFE_MINOR_VERSION, 1347
CFE_MISSION_REV, 1347
CFE_REVISION, 1347
CFE_SRC_VERSION, 1348
CFE_STR_HELPER, 1348
CFE_STR, 1348
CFE_VERSION_STRING, 1348
CheckErrCtr
CFE_ES_MemPoolStats, 698
ChildBuffer
FM_GlobalData_t, 902
ChildCmdCounter
FM_GlobalData_t, 903
FM_HousekeepingPkt_t, 911
ChildCmdErrCounter
FM_GlobalData_t, 903
FM_HousekeepingPkt_t, 912
ChildCmdWarnCounter
FM_GlobalData_t, 903
FM_HousekeepingPkt_t, 912
ChildCurrentCC
FM_GlobalData_t, 903
FM_HousekeepingPkt_t, 912

ChildPreviousCC
FM_GlobalData_t, 904
FM_HousekeepingPkt_t, 912
ChildQueue
FM_GlobalData_t, 904
ChildQueueCount
FM_GlobalData_t, 904
FM_HousekeepingPkt_t, 913
ChildQueueCountSem
FM_GlobalData_t, 904
ChildReadIndex
FM_GlobalData_t, 905
ChildSemaphore
FM_GlobalData_t, 905
ChildTaskID
FM_GlobalData_t, 905
ChildWriteIndex
FM_GlobalData_t, 905
ClockFlyState
CFE_TIME_DiagnosticTlm_Payload, 841
ClockSetState
CFE_TIME_DiagnosticTlm_Payload, 841
ClockSignal
CFE_TIME_DiagnosticTlm_Payload, 841
ClockSource
CFE_TIME_DiagnosticTlm_Payload, 841
ClockState
CFE_TIME_StateCmd_Payload, 865
ClockStateAPI
CFE_TIME_DiagnosticTlm_Payload, 842
CFE_TIME_HousekeepingTlm_Payload, 852
ClockStateFlags
CFE_TIME_DiagnosticTlm_Payload, 842
CFE_TIME_HousekeepingTlm_Payload, 852
CmdHeader
FM_ConcatCmd_t, 875
FM_CopyFileCmd_t, 877
FM_CreateDirCmd_t, 878
FM_DecompressCmd_t, 879
FM_DeleteAllCmd_t, 880
FM_DeleteDirCmd_t, 881
FM_DeleteFileCmd_t, 882
FM_GetDirFileCmd_t, 895
FM_GetDirPktCmd_t, 896
FM_GetFileInfoCmd_t, 898
FM_GetFreeSpaceCmd_t, 900
FM_GetOpenFilesCmd_t, 900
FM_HousekeepingCmd_t, 910
FM_MoveFileCmd_t, 915
FM_NoopCmd_t, 916
FM_RenameFileCmd_t, 919
FM_ResetCmd_t, 921
FM_SetPermCmd_t, 922
FM_SetTableStateCmd_t, 923
CmdPipe
FM_GlobalData_t, 906
code_address
OS_module_address_t, 934
code_size
OS_module_address_t, 934
CodeAddress
CFE_ES_AppInfo, 668
CodeSize
CFE_ES_AppInfo, 668
CommandCode
FM_ChildQueueEntry_t, 871
CommandCounter
CFE_ES_HousekeepingTlm_Payload, 688
CFE_EVS_HousekeepingTlm_Payload, 743
CFE_SB_HousekeepingTlm_Payload, 769
CFE_TBL_HousekeepingTlm_Payload, 814
CFE_TIME_HousekeepingTlm_Payload, 852
FM_GlobalData_t, 906
FM_HousekeepingPkt_t, 913
CommandErrCounter
FM_GlobalData_t, 906
FM_HousekeepingPkt_t, 913
CommandErrorCounter
CFE_ES_HousekeepingTlm_Payload, 688
CFE_EVS_HousekeepingTlm_Payload, 743
CFE_SB_HousekeepingTlm_Payload, 769
CFE_TBL_HousekeepingTlm_Payload, 814
CFE_TIME_HousekeepingTlm_Payload, 853
CommandHeader
CFE_ES_AppNameCmd, 673
CFE_ES_DeleteCDSCmd, 679
CFE_ES_DumpCDSRegistryCmd, 681
CFE_ES_FileNameCmd, 682
CFE_ES_NoArgsCmd, 701
CFE_ES_OverWriteSysLogCmd, 703
CFE_ES_ReloadAppCmd, 707
CFE_ES_RestartCmd, 708
CFE_ES_SendMemPoolStatsCmd, 710
CFE_ES_SetMaxPRCountCmd, 712
CFE_ES_SetPerfFilterMaskCmd, 714
CFE_ES_SetPerfTriggerMaskCmd, 716
CFE_ES_StartApp, 718
CFE_ES_StartPerfDataCmd, 722
CFE_ES_StopPerfDataCmd, 724
CFE_EVS_AppNameBitMaskCmd, 728
CFE_EVS_AppNameCmd, 730
CFE_EVS_AppNameEventIDCmd, 732
CFE_EVS_AppNameEventIDMaskCmd, 734
CFE_EVS_BitMaskCmd, 739
CFE_EVS_NoArgsCmd, 750
CFE_EVS_SetEventFormatModeCmd, 754
CFE_EVS_SetLogModeCmd, 756
CFE_EVS_WriteAppDataFileCmd, 758

CFE_EVS_WriteLogDataFileCmd, 759
CFE_SB_RouteCmd, 783
CFE_SB_WriteFileInfoCmd, 797
CFE_TBL_AbortLoadCmd, 799
CFE_TBL_ActivateCmd, 801
CFE_TBL_DeleteCDSCmd, 803
CFE_TBL_DumpCmd, 804
CFE_TBL_DumpRegistryCmd, 807
CFE_TBL_LoadCmd, 822
CFE_TBL_NoArgsCmd, 824
CFE_TBL_NotifyCmd, 825
CFE_TBL_SendRegistryCmd, 827
CFE_TBL_ValidateCmd, 835
CFE_TIME_NoArgsCmd, 857
CFE_TIME_OneHzAdjustmentCmd, 857
CFE_TIME_SetLeapSecondsCmd, 859
CFE_TIME_SetSignalCmd, 860
CFE_TIME_SetSourceCmd, 861
CFE_TIME_SetStateCmd, 862
CFE_TIME_TimeCmd, 867
CFE_TIME_ToneDataCmd, 868

common_types.h
EXTENSION, 1595
CompileTimeAssert, 1595, 1600, 1601
cpuaddr, 1596
cpudiff, 1597
cpusize, 1597
int16, 1597
int32, 1597
int64, 1597
int8, 1597
intptr, 1598
OS_ArgCallback_t, 1598
OS_PRINTF, 1595
OS_USED, 1595
OSAL_BLOCKCOUNT_C, 1595
OSAL_INDEX_C, 1596
OSAL_OBJTYPE_C, 1596
OSAL_SIZE_C, 1596
OSAL_STATUS_C, 1596
osal_blockcount_t, 1598
osal_id_t, 1598
osal_index_t, 1598
osal_objtype_t, 1599
osal_status_t, 1599
uint16, 1599
uint32, 1599
uint64, 1599
uint8, 1600

CompileTimeAssert
common_types.h, 1595, 1600, 1601

ContentType
CFE_FS_Header, 762

cpu1_msgids.h
CFE_ES_APP_TLM_MID, 1155
CFE_ES_CMD_MID, 1155
CFE_ES_HK_TLM_MID, 1155
CFE_ES_MEMSTATS_TLM_MID, 1155
CFE_ES_SEND_HK_MID, 1155
CFE_EVS_CMD_MID, 1155
CFE_EVS_HK_TLM_MID, 1155
CFE_EVS_LONG_EVENT_MSG_MID, 1156
CFE_EVS_SEND_HK_MID, 1156
CFE_EVS_SHORT_EVENT_MSG_MID, 1156
CFE_PLATFORM_CMD_MID_BASE_GLOB, 1156
CFE_PLATFORM_CMD_MID_BASE, 1156
CFE_PLATFORM_TLM_MID_BASE, 1157
CFE_SB_ALLSUBS_TLM_MID, 1157
CFE_SB_CMD_MID, 1157
CFE_SB_HK_TLM_MID, 1157
CFE_SB_ONESUB_TLM_MID, 1157
CFE_SB_SEND_HK_MID, 1158
CFE_SB_STATS_TLM_MID, 1158
CFE_SB_SUB_RPT_CTRL_MID, 1158
CFE_TBL_CMD_MID, 1158
CFE_TBL_HK_TLM_MID, 1158
CFE_TBL_REG_TLM_MID, 1158
CFE_TBL_SEND_HK_MID, 1159
CFE_TEST_CMD_MID, 1159
CFE_TEST_HK_TLM_MID, 1159
CFE_TIME_1HZ_CMD_MID, 1159
CFE_TIME_CMD_MID, 1159
CFE_TIME_DATA_CMD_MID, 1159
CFE_TIME_DIAG_TLM_MID, 1160
CFE_TIME_HK_TLM_MID, 1160
CFE_TIME_SEND_CMD_MID, 1160
CFE_TIME_SEND_HK_MID, 1160
CFE_TIME_TONE_CMD_MID, 1160

cpu1_platform_cfg.h
CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, 1164
CFE_PLATFORM_ENDIAN, 1165
CFE_PLATFORM_ES_APP_KILL_TIMEOUT, 1165
CFE_PLATFORM_ES_APP_SCAN_RATE, 1166
CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, 1166
CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩_01, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩_02, 1167
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩_03, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩_04, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩_05, 1168

CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_06, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_07, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_08, 1168
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_09, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_10, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_11, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_12, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_13, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_14, 1169
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_15, 1170
CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE↩
_16, 1170
CFE_PLATFORM_ES_CDS_SIZE, 1170
CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE,
1170
CFE_PLATFORM_ES_DEFAULT_CDS_REG_DU↩
MP_FILE, 1171
CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE,
1171
CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_↩
FILENAME, 1171
CFE_PLATFORM_ES_DEFAULT_POR_SYSLOG↩
_MODE, 1172
CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_↩
MODE, 1172
CFE_PLATFORM_ES_DEFAULT_STACK_SIZE,
1173
CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE,
1173
CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FI↩
LE, 1174
CFE_PLATFORM_ES_ER_LOG_ENTRIES, 1174
CFE_PLATFORM_ES_ER_LOG_MAX_CONTEX↩
T_SIZE, 1175
CFE_PLATFORM_ES_MAX_APPLICATIONS, 1175
CFE_PLATFORM_ES_MAX_BLOCK_SIZE, 1175
CFE_PLATFORM_ES_MAX_GEN_COUNTERS,
1176
CFE_PLATFORM_ES_MAX_LIBRARIES, 1176
CFE_PLATFORM_ES_MAX_MEMORY_POOLS,
1176
CFE_PLATFORM_ES_MAX_PROCESSOR_RES↩
ETS, 1177
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01,
1177
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02,
1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03,
1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04,
1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05,
1178
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11,
1179
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12,
1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13,
1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14,
1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15,
1180
CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16,
1180
CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE↩
_MIN, 1180
CFE_PLATFORM_ES_NONVOL_DISK_MOUNT_↩
STRING, 1181
CFE_PLATFORM_ES_NONVOL_STARTUP_FILE,
1181
CFE_PLATFORM_ES_OBJECT_TABLE_SIZE,
1181
CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY,
1182
CFE_PLATFORM_ES_PERF_CHILD_PRIORITY,
1182
CFE_PLATFORM_ES_PERF_CHILD_STACK_SI↩
ZE, 1182
CFE_PLATFORM_ES_PERF_DATA_BUFFER_SI↩
ZE, 1183
CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_↩
DLYS, 1183
CFE_PLATFORM_ES_PERF_FILTMASK_ALL,
1183
CFE_PLATFORM_ES_PERF_FILTMASK_INIT,
1184
CFE_PLATFORM_ES_PERF_FILTMASK_NONE,

1184
CFE_PLATFORM_ES_PERF_TRIGMASK_ALL, 1184
CFE_PLATFORM_ES_PERF_TRIGMASK_INIT, 1185
CFE_PLATFORM_ES_PERF_TRIGMASK_NONE, 1185
CFE_PLATFORM_ES_POOL_MAX_BUCKETS, 1185
CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING, 1186
CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS, 1186
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED, 1186
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE, 1187
CFE_PLATFORM_ES_RESET_AREA_SIZE, 1187
CFE_PLATFORM_ES_START_TASK_PRIORITY, 1188
CFE_PLATFORM_ES_START_TASK_STACK_SIZE, 1188
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC, 1189
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC, 1189
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, 1190
CFE_PLATFORM_ES_USER_RESERVED_SIZE, 1190
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE, 1191
CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE, 1191
CFE_PLATFORM_EVS_DEFAULT_LOG_FILE, 1192
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE, 1192
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE, 1192
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG, 1193
CFE_PLATFORM_EVS_LOG_MAX, 1193
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, 1193
CFE_PLATFORM_EVS_PORT_DEFAULT, 1194
CFE_PLATFORM_EVS_START_TASK_PRIORITY, 1194
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE, 1194
CFE_PLATFORM_SB_BUF_MEMORY_BYTES, 1195
CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME, 1195
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, 1196
CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME, 1196
CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME, 1197
CFE_PLATFORM_SB_FILTER_MASK1, 1197
CFE_PLATFORM_SB_FILTER_MASK2, 1197
CFE_PLATFORM_SB_FILTER_MASK3, 1198
CFE_PLATFORM_SB_FILTER_MASK4, 1198
CFE_PLATFORM_SB_FILTER_MASK5, 1198
CFE_PLATFORM_SB_FILTER_MASK6, 1198
CFE_PLATFORM_SB_FILTER_MASK7, 1198
CFE_PLATFORM_SB_FILTER_MASK8, 1198
CFE_PLATFORM_SB_FILTERED_EVENT1, 1199
CFE_PLATFORM_SB_FILTERED_EVENT2, 1199
CFE_PLATFORM_SB_FILTERED_EVENT3, 1199
CFE_PLATFORM_SB_FILTERED_EVENT4, 1199
CFE_PLATFORM_SB_FILTERED_EVENT5, 1199
CFE_PLATFORM_SB_FILTERED_EVENT6, 1200
CFE_PLATFORM_SB_FILTERED_EVENT7, 1200
CFE_PLATFORM_SB_FILTERED_EVENT8, 1200
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID, 1200
CFE_PLATFORM_SB_MAX_BLOCK_SIZE, 1200
CFE_PLATFORM_SB_MAX_DEST_PER_PKT, 1201
CFE_PLATFORM_SB_MAX_MSG_IDS, 1201
CFE_PLATFORM_SB_MAX_PIPES, 1201
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03, 1202
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09, 1203
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13, 1204

CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15, 1204
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16, 1205
CFE_PLATFORM_SB_START_TASK_PRIORITY, 1205
CFE_PLATFORM_SB_START_TASK_STACK_SI↩ ZE, 1205
CFE_PLATFORM_TBL_BUF_MEMORY_BYTES, 1205
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_↩ FILE, 1206
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES, 1206
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE, 1206
CFE_PLATFORM_TBL_MAX_NUM_HANDLES, 1207
CFE_PLATFORM_TBL_MAX_NUM_TABLES, 1207
CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS, 1207
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_↩ LOADS, 1208
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE, 1208
CFE_PLATFORM_TBL_START_TASK_PRIORITY, 1209
CFE_PLATFORM_TBL_START_TASK_STACK_↩ SIZE, 1209
CFE_PLATFORM_TBL_U32FROM4CHARS, 1209
CFE_PLATFORM_TBL_VALID_PRID_1, 1210
CFE_PLATFORM_TBL_VALID_PRID_2, 1210
CFE_PLATFORM_TBL_VALID_PRID_3, 1210
CFE_PLATFORM_TBL_VALID_PRID_4, 1210
CFE_PLATFORM_TBL_VALID_PRID_COUNT, 1211
CFE_PLATFORM_TBL_VALID_SCID_1, 1211
CFE_PLATFORM_TBL_VALID_SCID_2, 1211
CFE_PLATFORM_TBL_VALID_SCID_COUNT, 1212
CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY, 1212
CFE_PLATFORM_TIME_1HZ_TASK_STACK_SI↩ ZE, 1212
CFE_PLATFORM_TIME_CFG_CLIENT, 1212
CFE_PLATFORM_TIME_CFG_LATCH_FLY, 1212
CFE_PLATFORM_TIME_CFG_SERVER, 1213
CFE_PLATFORM_TIME_CFG_SIGNAL, 1213
CFE_PLATFORM_TIME_CFG_SOURCE, 1213
CFE_PLATFORM_TIME_CFG_SRC_GPS, 1214
CFE_PLATFORM_TIME_CFG_SRC_MET, 1214
CFE_PLATFORM_TIME_CFG_SRC_TIME, 1214
CFE_PLATFORM_TIME_CFG_START_FLY, 1215
CFE_PLATFORM_TIME_CFG_TONE_LIMIT, 1215
CFE_PLATFORM_TIME_CFG_VIRTUAL, 1215
CFE_PLATFORM_TIME_MAX_DELTA_SECS, 1216
CFE_PLATFORM_TIME_MAX_DELTA_SUBS, 1216
CFE_PLATFORM_TIME_MAX_LOCAL_SECS, 1217
CFE_PLATFORM_TIME_MAX_LOCAL_SUBS, 1217
CFE_PLATFORM_TIME_START_TASK_PRIORI↩ TY, 1217
CFE_PLATFORM_TIME_START_TASK_STACK_↩ SIZE, 1217
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY, 1218
CFE_PLATFORM_TIME_TONE_TASK_STACK_↩ SIZE, 1218
cpuaddr
common_types.h, 1596
cpudiff
common_types.h, 1597
cpusize
common_types.h, 1597
Crc
CFE_TBL_Info, 819
CFE_TBL_TblRegPacket_Payload, 831
CreatePipeErrorCounter
CFE_SB_HousekeepingTlm_Payload, 769
creator
OS_bin_sem_prop_t, 925
OS_count_sem_prop_t, 926
OS_mut_sem_prop_t, 937
OS_queue_prop_t, 938
OS_socket_prop_t, 941
OS_task_prop_t, 944
OS_timebase_prop_t, 947
OS_timer_prop_t, 948
Critical
CFE_TBL_Info, 819
CFE_TBL_TblRegPacket_Payload, 831
CurrentLatch
CFE_TIME_DiagnosticTlm_Payload, 842
CurrentMET
CFE_TIME_DiagnosticTlm_Payload, 842
CurrentQueueDepth
CFE_SB_PipeDepthStats, 777
CFE_SB_PipeInfoEntry, 779
CurrentTAI
CFE_TIME_DiagnosticTlm_Payload, 843
CurrentUTC
CFE_TIME_DiagnosticTlm_Payload, 843

data_address
OS_module_address_t, 935

data_size
OS_module_address_t, 935
DataAddress
CFE_ES_AppInfo, 668
DataFileName
CFE_ES_StopPerfCmd_Payload, 723
DataSize
CFE_ES_AppInfo, 669
DataStoreStatus
CFE_TIME_DiagnosticTlm_Payload, 843
DelayDirection
CFE_TIME_DiagnosticTlm_Payload, 843
Description
CFE_FS_FileWriteMetaData, 760
CFE_FS_Header, 763
CFE_TBL_FileDef, 810
DirEntries
FM_DirListFileStats_t, 885
DirListFileStats
FM_GlobalData_t, 906
DirListOffset
FM_ChildQueueEntry_t, 872
FM_GetDirPktCmd_t, 897
DirListPkt
FM_GlobalData_t, 907
DirName
FM_DirListFileStats_t, 885
FM_DirListPkt_t, 886
Directory
FM_CreateDirCmd_t, 878
FM_DeleteAllCmd_t, 880
FM_DeleteDirCmd_t, 881
FM_GetDirFileCmd_t, 895
FM_GetDirPktCmd_t, 897
DoubleBuffered
CFE_TBL_Info, 820
CFE_TBL_TblRegPacket_Payload, 831
DumpFilename
CFE_ES_DumpCDSRegistryCmd_Payload, 682
CFE_TBL_DumpCmd_Payload, 806
CFE_TBL_DumpRegistryCmd_Payload, 808
DumpOnly
CFE_TBL_Info, 820
CFE_TBL_TblRegPacket_Payload, 831
DuplicateSubscriptionsCounter
CFE_SB_HousekeepingTlm_Payload, 769

ERLogEntries
CFE_ES_HousekeepingTlm_Payload, 688
ERLogIndex
CFE_ES_HousekeepingTlm_Payload, 689
Entries
CFE_SB_AllSubscriptionsTlm_Payload, 766
Entry
CFE_SB_AllSubscriptionsTlm_Payload, 766
entry_point
OS_module_prop_t, 936
EntryName
FM_DirListEntry_t, 883
EntryPoint
CFE_ES_AppInfo, 669
EntrySize
FM_DirListEntry_t, 884
EventID
CFE_EVS_AppNameEventIDCmd_Payload, 733
CFE_EVS_AppNameEventIDMaskCmd_Payload, 735
CFE_EVS_BinFilter, 738
CFE_EVS_PacketID, 751
EventType
CFE_EVS_PacketID, 751
ExceptionAction
CFE_ES_AppInfo, 669
CFE_ES_StartAppCmd_Payload, 720
ExecutionCounter
CFE_ES_AppInfo, 669
CFE_ES_TaskInfo, 725

FM_APP_NAME
CFS File Manager Platform Configuration, 135
FM_APP_PIPE_DEPTH
CFS File Manager Platform Configuration, 135
FM_APP_PIPE_NAME
CFS File Manager Platform Configuration, 135
FM_APPMAIN_PERF_ID
CFS File Manager Mission Configuration, 131
FM_AcquireTablePointers
fm_tbl.c, 1133
fm_tbl.h, 1138
FM_AppInit
fm_app.c, 952
fm_app.h, 963
FM_AppMain
fm_app.c, 953
fm_app.h, 964
FM_AppendPathSep
fm_cmd_utils.c, 1029
fm_cmd_utils.h, 1049
FM_CC_ERR_EID
CFS File Manager Event IDs, 158
FM_CHILD_BROKEN_EID_OFFSET
CFS File Manager Event IDs, 158
FM_CHILD_DISABLED_EID_OFFSET
CFS File Manager Event IDs, 158
FM_CHILD_EXE_ERR_EID
CFS File Manager Event IDs, 159
FM_CHILD_FILE_BLOCK_SIZE
CFS File Manager Platform Configuration, 136

FM_CHILD_FILE_LOOP_COUNT
CFS File Manager Platform Configuration, 136
FM_CHILD_FILE_SLEEP_MS
CFS File Manager Platform Configuration, 137
FM_CHILD_INIT_CREATE_ERR_EID
CFS File Manager Event IDs, 159
FM_CHILD_INIT_EID
CFS File Manager Event IDs, 159
FM_CHILD_INIT_QSEM_ERR_EID
CFS File Manager Event IDs, 160
FM_CHILD_INIT_SEM_ERR_EID
CFS File Manager Event IDs, 160
FM_CHILD_NUM_OFFSETS
CFS File Manager Event IDs, 161
FM_CHILD_Q_FULL_EID_OFFSET
CFS File Manager Event IDs, 161
FM_CHILD_QUEUE_DEPTH
CFS File Manager Platform Configuration, 137
FM_CHILD_SEM_NAME
CFS File Manager Platform Configuration, 137
FM_CHILD_STAT_SLEEP_FILECOUNT
CFS File Manager Platform Configuration, 138
FM_CHILD_STAT_SLEEP_MS
CFS File Manager Platform Configuration, 138
FM_CHILD_TASK_NAME
CFS File Manager Platform Configuration, 139
FM_CHILD_TASK_PERF_ID
CFS File Manager Mission Configuration, 131
FM_CHILD_TASK_PRIORITY
CFS File Manager Platform Configuration, 139
FM_CHILD_TASK_STACK_SIZE
CFS File Manager Platform Configuration, 140
FM_CHILD_TERM_EMPTYQ_ERR_EID
CFS File Manager Event IDs, 161
FM_CHILD_TERM_QIDX_ERR_EID
CFS File Manager Event IDs, 162
FM_CHILD_TERM_SEM_ERR_EID
CFS File Manager Event IDs, 162
FM_CMD_MID
CFS File Manager Command Message IDs, 132
FM_CONCAT_CHILD_BASE_EID
CFS File Manager Event IDs, 163
FM_CONCAT_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 163
FM_CONCAT_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 164
FM_CONCAT_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 164
FM_CONCAT_CMD_EID
CFS File Manager Event IDs, 165
FM_CONCAT_CC
CFS File Manager Command Codes, 278
FM_CONCAT_OPEN_SRC2_ERR_EID
CFS File Manager Event IDs, 165
FM_CONCAT_OPEN_TGT_ERR_EID
CFS File Manager Event IDs, 166
FM_CONCAT_OSCPY_ERR_EID
CFS File Manager Event IDs, 166
FM_CONCAT_OSRD_ERR_EID
CFS File Manager Event IDs, 167
FM_CONCAT_OSWR_ERR_EID
CFS File Manager Event IDs, 167
FM_CONCAT_PKT_ERR_EID
CFS File Manager Event IDs, 168
FM_CONCAT_SRC1_BASE_EID
CFS File Manager Event IDs, 168
FM_CONCAT_SRC1_DNE_ERR_EID
CFS File Manager Event IDs, 169
FM_CONCAT_SRC1_INVALID_ERR_EID
CFS File Manager Event IDs, 169
FM_CONCAT_SRC1_ISDIR_ERR_EID
CFS File Manager Event IDs, 170
FM_CONCAT_SRC1_OPEN_ERR_EID
CFS File Manager Event IDs, 170
FM_CONCAT_SRC2_BASE_EID
CFS File Manager Event IDs, 171
FM_CONCAT_SRC2_DNE_ERR_EID
CFS File Manager Event IDs, 171
FM_CONCAT_SRC2_INVALID_ERR_EID
CFS File Manager Event IDs, 172
FM_CONCAT_SRC2_ISDIR_ERR_EID
CFS File Manager Event IDs, 172
FM_CONCAT_SRC2_OPEN_ERR_EID
CFS File Manager Event IDs, 173
FM_CONCAT_TGT_BASE_EID
CFS File Manager Event IDs, 173
FM_CONCAT_TGT_EXIST_ERR_EID
CFS File Manager Event IDs, 174
FM_CONCAT_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 174
FM_CONCAT_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 175
FM_COPY_CHILD_BASE_EID
CFS File Manager Event IDs, 175
FM_COPY_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 176
FM_COPY_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 176
FM_COPY_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 177
FM_COPY_CMD_EID
CFS File Manager Event IDs, 177
FM_COPY_CC
CFS File Manager Command Codes, 280
FM_COPY_OS_ERR_EID
CFS File Manager Event IDs, 178
FM_COPY_OVR_ERR_EID
CFS File Manager Event IDs, 178

FM_COPY_PKT_ERR_EID
CFS File Manager Event IDs, 179
FM_COPY_SRC_BASE_EID
CFS File Manager Event IDs, 179
FM_COPY_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 180
FM_COPY_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 180
FM_COPY_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 181
FM_COPY_SRC_UNKNOWN_ERR_EID
CFS File Manager Event IDs, 181
FM_COPY_TGT_BASE_EID
CFS File Manager Event IDs, 182
FM_COPY_TGT_EXIST_ERR_EID
CFS File Manager Event IDs, 182
FM_COPY_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 183
FM_COPY_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 183
FM_COPY_TGT_ISOPEN_ERR_EID
CFS File Manager Event IDs, 184
FM_COPY_TGT_UNKNOWN_ERR_EID
CFS File Manager Event IDs, 184
FM_CREATE_DIR_CHILD_BASE_EID
CFS File Manager Event IDs, 185
FM_CREATE_DIR_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 185
FM_CREATE_DIR_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 186
FM_CREATE_DIR_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 186
FM_CREATE_DIR_CMD_EID
CFS File Manager Event IDs, 187
FM_CREATE_DIR_CC
CFS File Manager Command Codes, 281
FM_CREATE_DIR_OS_ERR_EID
CFS File Manager Event IDs, 187
FM_CREATE_DIR_PKT_ERR_EID
CFS File Manager Event IDs, 188
FM_CREATE_DIR_SRC_BASE_EID
CFS File Manager Event IDs, 188
FM_CREATE_DIR_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 189
FM_CREATE_DIR_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 189
FM_CREATE_DIR_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 190
FM_ChildConcatCmd
fm_child.c, 974
fm_child.h, 1001
FM_ChildCopyCmd
fm_child.c, 975
fm_child.h, 1003
FM_ChildCreateDirCmd
fm_child.c, 976
fm_child.h, 1004
FM_ChildDeleteAllCmd
fm_child.c, 977
fm_child.h, 1005
FM_ChildDeleteCmd
fm_child.c, 978
fm_child.h, 1006
FM_ChildDeleteDirCmd
fm_child.c, 979
fm_child.h, 1007
FM_ChildDirListFileCmd
fm_child.c, 981
fm_child.h, 1009
FM_ChildDirListFileInit
fm_child.c, 982
fm_child.h, 1010
FM_ChildDirListFileLoop
fm_child.c, 984
fm_child.h, 1012
FM_ChildDirListPktCmd
fm_child.c, 985
fm_child.h, 1013
FM_ChildFileInfoCmd
fm_child.c, 986
fm_child.h, 1014
FM_ChildInit
fm_child.c, 988
fm_child.h, 1016
FM_ChildLoop
fm_child.c, 990
fm_child.h, 1018
FM_ChildMoveCmd
fm_child.c, 991
fm_child.h, 1019
FM_ChildProcess
fm_child.c, 992
fm_child.h, 1020
FM_ChildQueueEntry_t, 871
CommandCode, 871
DirListOffset, 872
FileInfoCRC, 872
FileInfoSize, 872
FileInfoState, 872
FileInfoTime, 873
GetSizeTimeMode, 873
Mode, 873
Source1, 873
Source2, 874
Target, 874
FM_ChildRenameCmd
fm_child.c, 994
fm_child.h, 1022

FM_ChildSetPermissionsCmd
fm_child.c, 995
fm_child.h, 1023
FM_ChildSizeTimeMode
fm_child.c, 996
fm_child.h, 1024
FM_ChildSleepStat
fm_child.c, 997
fm_child.h, 1025
FM_ChildTask
fm_child.c, 998
fm_child.h, 1026
FM_ConcatCmd_t, 874
CmdHeader, 875
Source1, 875
Source2, 875
Target, 876
FM_ConcatFilesCmd
fm_cmds.c, 1067
fm_cmds.h, 1091
FM_CopyFileCmd
fm_cmds.c, 1069
fm_cmds.h, 1092
FM_CopyFileCmd_t, 876
CmdHeader, 877
Overwrite, 877
Source, 877
Target, 877
FM_CreateDirCmd_t, 878
CmdHeader, 878
Directory, 878
FM_CreateDirectoryCmd
fm_cmds.c, 1070
fm_cmds.h, 1094
FM_DECOM_CFE_ERR_EID
CFS File Manager Event IDs, 190
FM_DECOM_CHILD_BASE_EID
CFS File Manager Event IDs, 191
FM_DECOM_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 191
FM_DECOM_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 192
FM_DECOM_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 192
FM_DECOM_CMD_EID
CFS File Manager Event IDs, 193
FM_DECOM_PKT_ERR_EID
CFS File Manager Event IDs, 193
FM_DECOM_SRC_BASE_EID
CFS File Manager Event IDs, 194
FM_DECOM_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 194
FM_DECOM_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 195
FM_DECOM_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 195
FM_DECOM_SRC_OPEN_ERR_EID
CFS File Manager Event IDs, 196
FM_DECOM_TGT_BASE_EID
CFS File Manager Event IDs, 196
FM_DECOM_TGT_EXIST_ERR_EID
CFS File Manager Event IDs, 197
FM_DECOM_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 197
FM_DECOM_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 198
FM_DECOMPRESS_CC
CFS File Manager Command Codes, 282
FM_DELETE_ALL_CHILD_BASE_EID
CFS File Manager Event IDs, 198
FM_DELETE_ALL_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 199
FM_DELETE_ALL_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 199
FM_DELETE_ALL_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 200
FM_DELETE_ALL_CMD_EID
CFS File Manager Event IDs, 200
FM_DELETE_ALL_CC
CFS File Manager Command Codes, 284
FM_DELETE_ALL_FILES_ND_WARNING_EID
CFS File Manager Event IDs, 201
FM_DELETE_ALL_OS_ERR_EID
CFS File Manager Event IDs, 201
FM_DELETE_ALL_PKT_ERR_EID
CFS File Manager Event IDs, 202
FM_DELETE_ALL_SKIP_WARNING_EID
CFS File Manager Event IDs, 202
FM_DELETE_ALL_SRC_BASE_EID
CFS File Manager Event IDs, 203
FM_DELETE_ALL_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 203
FM_DELETE_ALL_SRC_FILE_ERR_EID
CFS File Manager Event IDs, 204
FM_DELETE_ALL_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 204
FM_DELETE_CHILD_BASE_EID
CFS File Manager Event IDs, 205
FM_DELETE_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 205
FM_DELETE_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 206
FM_DELETE_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 206
FM_DELETE_CMD_EID
CFS File Manager Event IDs, 207
FM_DELETE_CC
CFS File Manager Command Codes, 285

FM_DELETE_DIR_CHILD_BASE_EID
CFS File Manager Event IDs, 207
FM_DELETE_DIR_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 208
FM_DELETE_DIR_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 208
FM_DELETE_DIR_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 209
FM_DELETE_DIR_CMD_EID
CFS File Manager Event IDs, 209
FM_DELETE_DIR_CC
CFS File Manager Command Codes, 287
FM_DELETE_DIR_EMPTY_ERR_EID
CFS File Manager Event IDs, 210
FM_DELETE_DIR_PKT_ERR_EID
CFS File Manager Event IDs, 210
FM_DELETE_DIR_SRC_BASE_EID
CFS File Manager Event IDs, 211
FM_DELETE_DIR_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 211
FM_DELETE_DIR_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 212
FM_DELETE_INT_CC
CFS File Manager Command Codes, 288
FM_DELETE_OPENDIR_OS_ERR_EID
CFS File Manager Event IDs, 212
FM_DELETE_OS_ERR_EID
CFS File Manager Event IDs, 213
FM_DELETE_PKT_ERR_EID
CFS File Manager Event IDs, 213
FM_DELETE_RMDIR_OS_ERR_EID
CFS File Manager Event IDs, 214
FM_DELETE_SRC_BASE_EID
CFS File Manager Event IDs, 214
FM_DELETE_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 215
FM_DELETE_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 215
FM_DELETE_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 216
FM_DELETE_SRC_OPEN_ERR_EID
CFS File Manager Event IDs, 216
FM_DIR_LIST_FILE_DEFNAME
CFS File Manager Platform Configuration, 140
FM_DIR_LIST_FILE_ENTRIES
CFS File Manager Platform Configuration, 141
FM_DIR_LIST_FILE_SUBTYPE
CFS File Manager Platform Configuration, 141
FM_DIR_LIST_PKT_ENTRIES
CFS File Manager Platform Configuration, 142
FM_DIR_LIST_TLM_MID
CFS File Manager Telemetry Message IDs, 133
FM_DecompressCmd_t, 879
CmdHeader, 879
Source, 879
Target, 879
FM_DeleteAllCmd_t, 880
CmdHeader, 880
Directory, 880
FM_DeleteAllFilesCmd
fm_cmds.c, 1071
fm_cmds.h, 1096
FM_DeleteDirCmd_t, 881
CmdHeader, 881
Directory, 881
FM_DeleteDirectoryCmd
fm_cmds.c, 1073
fm_cmds.h, 1097
FM_DeleteFileCmd
fm_cmds.c, 1074
fm_cmds.h, 1098
FM_DeleteFileCmd_t, 882
CmdHeader, 882
Filename, 882
FM_DirListEntry_t, 883
EntryName, 883
EntrySize, 884
Mode, 884
ModifyTime, 884
FM_DirListFileStats_t, 885
DirEntries, 885
DirName, 885
FileEntries, 885
FM_DirListPkt_t, 886
DirName, 886
FileList, 887
FirstFile, 887
PacketFiles, 887
TlmHeader, 887
TotalFiles, 888
FM_EXIT_ERR_EID
CFS File Manager Event IDs, 217
FM_FILE_INFO_CHILD_BASE_EID
CFS File Manager Event IDs, 217
FM_FILE_INFO_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 218
FM_FILE_INFO_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 218
FM_FILE_INFO_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 219
FM_FILE_INFO_TLM_MID
CFS File Manager Telemetry Message IDs, 133
FM_FNAME_DNE_EID_OFFSET
CFS File Manager Event IDs, 219
FM_FNAME_EXIST_EID_OFFSET
CFS File Manager Event IDs, 220
FM_FNAME_INVALID_EID_OFFSET
CFS File Manager Event IDs, 220

FM_FNAME_ISCLOSED_EID_OFFSET
CFS File Manager Event IDs, 220
FM_FNAME_ISDIR_EID_OFFSET
CFS File Manager Event IDs, 220
FM_FNAME_ISFILE_EID_OFFSET
CFS File Manager Event IDs, 220
FM_FNAME_ISOPEN_EID_OFFSET
CFS File Manager Event IDs, 221
FM_FNAME_NUM_OFFSETS
CFS File Manager Event IDs, 221
FM_FNAME_UNKNOWN_EID_OFFSET
CFS File Manager Event IDs, 221
FM_FREE_SPACE_TLM_MID
CFS File Manager Telemetry Message IDs, 133
FM_FileInfoPkt_t, 888
CRC_Computed, 889
CRC, 889
FileSize, 889
FileStatus, 890
Filename, 889
LastModifiedTime, 890
Mode, 890
Spare, 890
TlmHeader, 891
FM_FreeSpacePkt_t, 891
FileSys, 891
TlmHeader, 892
FM_FreeSpacePktEntry_t, 892
FreeSpace, 893
Name, 893
FM_FreeSpaceTable
fm_freespace.c, 1144
FM_FreeSpaceTable_t, 893
FileSys, 894
FM_GET_DIR_FILE_CHILD_BASE_EID
CFS File Manager Event IDs, 221
FM_GET_DIR_FILE_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 221
FM_GET_DIR_FILE_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 222
FM_GET_DIR_FILE_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 222
FM_GET_DIR_FILE_CMD_EID
CFS File Manager Event IDs, 223
FM_GET_DIR_FILE_CC
CFS File Manager Command Codes, 288
FM_GET_DIR_FILE_OSCREAT_ERR_EID
CFS File Manager Event IDs, 223
FM_GET_DIR_FILE_OSOPENDIR_ERR_EID
CFS File Manager Event IDs, 224
FM_GET_DIR_FILE_PKT_ERR_EID
CFS File Manager Event IDs, 224
FM_GET_DIR_FILE_SRC_BASE_EID
CFS File Manager Event IDs, 225
FM_GET_DIR_FILE_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 225
FM_GET_DIR_FILE_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 226
FM_GET_DIR_FILE_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 226
FM_GET_DIR_FILE_TGT_BASE_EID
CFS File Manager Event IDs, 227
FM_GET_DIR_FILE_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 227
FM_GET_DIR_FILE_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 228
FM_GET_DIR_FILE_UPSTATS_ERR_EID
CFS File Manager Event IDs, 228
FM_GET_DIR_FILE_WARNING_EID
CFS File Manager Event IDs, 229
FM_GET_DIR_FILE_WRBLANK_ERR_EID
CFS File Manager Event IDs, 229
FM_GET_DIR_FILE_WRENTRY_ERR_EID
CFS File Manager Event IDs, 230
FM_GET_DIR_FILE_WRHDR_ERR_EID
CFS File Manager Event IDs, 230
FM_GET_DIR_PKT_CHILD_BASE_EID
CFS File Manager Event IDs, 231
FM_GET_DIR_PKT_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 231
FM_GET_DIR_PKT_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 232
FM_GET_DIR_PKT_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 232
FM_GET_DIR_PKT_CMD_EID
CFS File Manager Event IDs, 233
FM_GET_DIR_PKT_CC
CFS File Manager Command Codes, 290
FM_GET_DIR_PKT_OS_ERR_EID
CFS File Manager Event IDs, 233
FM_GET_DIR_PKT_PKT_ERR_EID
CFS File Manager Event IDs, 234
FM_GET_DIR_PKT_SRC_BASE_EID
CFS File Manager Event IDs, 234
FM_GET_DIR_PKT_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 235
FM_GET_DIR_PKT_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 235
FM_GET_DIR_PKT_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 236
FM_GET_DIR_PKT_WARNING_EID
CFS File Manager Event IDs, 236
FM_GET_FILE_INFO_CMD_EID
CFS File Manager Event IDs, 237
FM_GET_FILE_INFO_CC
CFS File Manager Command Codes, 292
FM_GET_FILE_INFO_OPEN_ERR_EID
CFS File Manager Event IDs, 237

FM_GET_FILE_INFO_PKT_ERR_EID
CFS File Manager Event IDs, 238
FM_GET_FILE_INFO_READ_WARNING_EID
CFS File Manager Event IDs, 238
FM_GET_FILE_INFO_SRC_ERR_EID
CFS File Manager Event IDs, 239
FM_GET_FILE_INFO_STATE_WARNING_EID
CFS File Manager Event IDs, 239
FM_GET_FILE_INFO_TYPE_WARNING_EID
CFS File Manager Event IDs, 240
FM_GET_FREE_SPACE_CMD_EID
CFS File Manager Event IDs, 240
FM_GET_FREE_SPACE_CC
CFS File Manager Command Codes, 293
FM_GET_FREE_SPACE_PKT_ERR_EID
CFS File Manager Event IDs, 241
FM_GET_FREE_SPACE_TBL_ERR_EID
CFS File Manager Event IDs, 241
FM_GET_OPEN_FILES_CMD_EID
CFS File Manager Event IDs, 242
FM_GET_OPEN_FILES_CC
CFS File Manager Command Codes, 294
FM_GET_OPEN_FILES_PKT_ERR_EID
CFS File Manager Event IDs, 242
FM_GetDirFileCmd_t, 894
CmdHeader, 895
Directory, 895
Filename, 895
GetSizeTimeMode, 895
Spare01, 895
FM_GetDirListFileCmd
fm_cmds.c, 1075
fm_cmds.h, 1100
FM_GetDirListPktCmd
fm_cmds.c, 1077
fm_cmds.h, 1101
FM_GetDirPktCmd_t, 896
CmdHeader, 896
DirListOffset, 897
Directory, 897
GetSizeTimeMode, 897
Spare01, 897
FM_GetFileInfoCmd
fm_cmds.c, 1078
fm_cmds.h, 1103
FM_GetFileInfoCmd_t, 898
CmdHeader, 898
FileInfoCRC, 898
Filename, 899
FM_GetFilenameState
fm_cmd_utils.c, 1030
fm_cmd_utils.h, 1050
FM_GetFreeSpaceCmd
fm_cmds.c, 1080
fm_cmds.h, 1104
FM_GetFreeSpaceCmd_t, 899
CmdHeader, 900
FM_GetOpenFilesCmd
fm_cmds.c, 1081
fm_cmds.h, 1105
FM_GetOpenFilesCmd_t, 900
CmdHeader, 900
FM_GetOpenFilesData
fm_cmd_utils.c, 1031
fm_cmd_utils.h, 1051
FM_GlobalData
fm_app.c, 962
fm_msg.h, 1131
FM_GlobalData_t, 901
ChildBuffer, 902
ChildCmdCounter, 903
ChildCmdErrCounter, 903
ChildCmdWarnCounter, 903
ChildCurrentCC, 903
ChildPreviousCC, 904
ChildQueue, 904
ChildQueueCount, 904
ChildQueueCountSem, 904
ChildReadIndex, 905
ChildSemaphore, 905
ChildTaskID, 905
ChildWriteIndex, 905
CmdPipe, 906
CommandCounter, 906
CommandErrCounter, 906
DirListFileStats, 906
DirListPkt, 907
FileInfoPkt, 907
FileStatMode, 907
FileStatSize, 907
FileStatTime, 908
FreeSpacePkt, 908
FreeSpaceTableHandle, 908
FreeSpaceTablePtr, 908
HousekeepingPkt, 909
OpenFilesPkt, 909
Spare8a, 909
Spare8b, 909
FM_HK_REQ_ERR_EID
CFS File Manager Event IDs, 243
FM_HK_TLM_MID
CFS File Manager Telemetry Message IDs, 133
FM_HousekeepingCmd_t, 910
CmdHeader, 910
FM_HousekeepingPkt_t, 911
ChildCmdCounter, 911
ChildCmdErrCounter, 912
ChildCmdWarnCounter, 912

ChildCurrentCC, 912
ChildPreviousCC, 912
ChildQueueCount, 913
CommandCounter, 913
CommandErrCounter, 913
NumOpenFiles, 913
Spare, 914
TlmHeader, 914
FM_IGNORE_CRC
fm_defs.h, 1115
FM_InvokeChildTask
fm_cmd_utils.c, 1032
fm_cmd_utils.h, 1052
FM_IsValidCmdPktLength
fm_cmd_utils.c, 1033
fm_cmd_utils.h, 1053
FM_MAJOR_VERSION
CFS File Manager Version, 301
FM_MID_ERR_EID
CFS File Manager Event IDs, 243
FM_MINOR_VERSION
CFS File Manager Version, 301
FM_MISSION_REV
CFS File Manager Platform Configuration, 142
FM_MOVE_CHILD_BASE_EID
CFS File Manager Event IDs, 244
FM_MOVE_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 244
FM_MOVE_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 245
FM_MOVE_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 245
FM_MOVE_CMD_EID
CFS File Manager Event IDs, 246
FM_MOVE_CC
CFS File Manager Command Codes, 294
FM_MOVE_OS_ERR_EID
CFS File Manager Event IDs, 246
FM_MOVE_OVR_ERR_EID
CFS File Manager Event IDs, 247
FM_MOVE_PKT_ERR_EID
CFS File Manager Event IDs, 247
FM_MOVE_SRC_BASE_EID
CFS File Manager Event IDs, 248
FM_MOVE_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 248
FM_MOVE_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 249
FM_MOVE_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 249
FM_MOVE_TGT_BASE_EID
CFS File Manager Event IDs, 250
FM_MOVE_TGT_EXIST_ERR_EID
CFS File Manager Event IDs, 250
FM_MOVE_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 251
FM_MOVE_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 251
FM_MOVE_TGT_ISOPEN_ERR_EID
CFS File Manager Event IDs, 252
FM_MOVE_TGT_UNKNOWN_ERR_EID
CFS File Manager Event IDs, 252
FM_MoveFileCmd
fm_cmds.c, 1082
fm_cmds.h, 1107
FM_MoveFileCmd_t, 914
CmdHeader, 915
Overwrite, 915
Source, 915
Target, 915
FM_NAME_IS_DIRECTORY
fm_defs.h, 1115
FM_NAME_IS_FILE_CLOSED
fm_defs.h, 1115
FM_NAME_IS_FILE_OPEN
fm_defs.h, 1115
FM_NAME_IS_INVALID
fm_defs.h, 1115
FM_NAME_IS_NOT_IN_USE
fm_defs.h, 1116
FM_NOOP_CMD_EID
CFS File Manager Event IDs, 253
FM_NOOP_CC
CFS File Manager Command Codes, 296
FM_NOOP_PKT_ERR_EID
CFS File Manager Event IDs, 253
FM_NoopCmd
fm_cmds.c, 1084
fm_cmds.h, 1108
FM_NoopCmd_t, 916
CmdHeader, 916
FM_OPEN_FILES_TLM_MID
CFS File Manager Telemetry Message IDs, 133
FM_OS_SYS_STAT_ERR_EID
CFS File Manager Event IDs, 254
FM_OpenFilesEntry_t, 917
AppName, 917
LogicalName, 917
FM_OpenFilesPkt_t, 918
NumOpenFiles, 918
OpenFilesList, 918
TlmHeader, 918
FM_PARENT_DIRECTORY
fm_defs.h, 1116
FM_ProcessCmd
fm_app.c, 955
fm_app.h, 966
FM_ProcessPkt

fm_app.c, 958
fm_app.h, 968
FM_QUEUE_SEM_NAME
fm_child.c, 973
FM_RENAME_CHILD_BASE_EID
CFS File Manager Event IDs, 254
FM_RENAME_CHILD_BROKEN_ERR_EID
CFS File Manager Event IDs, 255
FM_RENAME_CHILD_DISABLED_ERR_EID
CFS File Manager Event IDs, 255
FM_RENAME_CHILD_FULL_ERR_EID
CFS File Manager Event IDs, 256
FM_RENAME_CMD_EID
CFS File Manager Event IDs, 256
FM_RENAME_CC
CFS File Manager Command Codes, 296
FM_RENAME_OS_ERR_EID
CFS File Manager Event IDs, 257
FM_RENAME_OVR_ERR_EID
CFS File Manager Event IDs, 257
FM_RENAME_PKT_ERR_EID
CFS File Manager Event IDs, 258
FM_RENAME_SRC_BASE_EID
CFS File Manager Event IDs, 258
FM_RENAME_SRC_DNE_ERR_EID
CFS File Manager Event IDs, 259
FM_RENAME_SRC_INVALID_ERR_EID
CFS File Manager Event IDs, 259
FM_RENAME_SRC_ISDIR_ERR_EID
CFS File Manager Event IDs, 260
FM_RENAME_TGT_BASE_EID
CFS File Manager Event IDs, 260
FM_RENAME_TGT_EXIST_ERR_EID
CFS File Manager Event IDs, 261
FM_RENAME_TGT_INVALID_ERR_EID
CFS File Manager Event IDs, 261
FM_RENAME_TGT_ISDIR_ERR_EID
CFS File Manager Event IDs, 262
FM_RENAME_TGT_ISOPEN_ERR_EID
CFS File Manager Event IDs, 262
FM_RENAME_TGT_UNKNOWN_ERR_EID
CFS File Manager Event IDs, 263
FM_RESET_CMD_EID
CFS File Manager Event IDs, 263
FM_RESET_CC
CFS File Manager Command Codes, 298
FM_RESET_PKT_ERR_EID
CFS File Manager Event IDs, 264
FM_REVISION
CFS File Manager Version, 301
FM_ReleaseTablePointers
fm_tbl.c, 1134
fm_tbl.h, 1139
FM_RenameFileCmd
fm_cmds.c, 1085
fm_cmds.h, 1109
FM_RenameFileCmd_t, 919
CmdHeader, 919
Source, 920
Target, 920
FM_ReportHK
fm_app.c, 960
fm_app.h, 970
FM_ResetCmd_t, 920
CmdHeader, 921
FM_ResetCountersCmd
fm_cmds.c, 1086
fm_cmds.h, 1110
FM_SB_RECEIVE_ERR_EID
CFS File Manager Event IDs, 264
FM_SB_RECEIVE_NULL_PTR_ERR_EID
CFS File Manager Event IDs, 265
FM_SB_TIMEOUT
fm_app.h, 963
FM_SEND_HK_MID
CFS File Manager Command Message IDs, 132
FM_SET_FILE_PERM_CC
CFS File Manager Command Codes, 298
FM_SET_PERM_CMD_EID
CFS File Manager Event IDs, 265
FM_SET_PERM_ERR_EID
CFS File Manager Event IDs, 266
FM_SET_PERM_OS_ERR_EID
CFS File Manager Event IDs, 266
FM_SET_TABLE_STATE_ARG_IDX_ERR_EID
CFS File Manager Event IDs, 267
FM_SET_TABLE_STATE_ARG_STATE_ERR_EID
CFS File Manager Event IDs, 267
FM_SET_TABLE_STATE_CMD_EID
CFS File Manager Event IDs, 268
FM_SET_TABLE_STATE_CC
CFS File Manager Command Codes, 299
FM_SET_TABLE_STATE_PKT_ERR_EID
CFS File Manager Event IDs, 268
FM_SET_TABLE_STATE_TBL_ERR_EID
CFS File Manager Event IDs, 269
FM_SET_TABLE_STATE_UNUSED_ERR_EID
CFS File Manager Event IDs, 269
FM_STARTUP_CREAT_PIPE_ERR_EID
CFS File Manager Event IDs, 270
FM_STARTUP_EID
CFS File Manager Event IDs, 270
FM_STARTUP_EVENTS_ERR_EID
CFS File Manager Event IDs, 271
FM_STARTUP_SUBSCRIB_GCMD_ERR_EID
CFS File Manager Event IDs, 271
FM_STARTUP_SUBSCRIB_HK_ERR_EID
CFS File Manager Event IDs, 272

FM_STARTUP_TABLE_INIT_ERR_EID
CFS File Manager Event IDs, 272
FM_SetPermCmd_t, 921
CmdHeader, 922
FileName, 922
Mode, 922
FM_SetPermissionsCmd
fm_cmds.c, 1087
fm_cmds.h, 1112
FM_SetTableStateCmd
fm_cmds.c, 1088
fm_cmds.h, 1113
FM_SetTableStateCmd_t, 922
CmdHeader, 923
TableEntryIndex, 923
TableEntryState, 923
FM_TABLE_CFE_NAME
CFS File Manager Platform Configuration, 143
FM_TABLE_DEF_DESC
CFS File Manager Platform Configuration, 143
FM_TABLE_DEF_NAME
CFS File Manager Platform Configuration, 144
FM_TABLE_ENTRY_COUNT
CFS File Manager Platform Configuration, 144
FM_TABLE_ENTRY_DISABLED
fm_defs.h, 1116
FM_TABLE_ENTRY_ENABLED
fm_defs.h, 1116
FM_TABLE_ENTRY_UNUSED
fm_defs.h, 1116
FM_TABLE_FILENAME
CFS File Manager Platform Configuration, 144
FM_TABLE_VALIDATION_ERR
CFS File Manager Platform Configuration, 145
FM_TABLE_VERIFY_BAD_STATE_ERR_EID
CFS File Manager Event IDs, 273
FM_TABLE_VERIFY_EID
CFS File Manager Event IDs, 273
FM_TABLE_VERIFY_EMPTY_ERR_EID
CFS File Manager Event IDs, 274
FM_TABLE_VERIFY_NULL_PTR_ERR_EID
CFS File Manager Event IDs, 274
FM_TABLE_VERIFY_TOOLONG_ERR_EID
CFS File Manager Event IDs, 275
FM_THIS_DIRECTORY
fm_defs.h, 1117
FM_TableEntry_t, 924
Name, 924
State, 924
FM_TableInit
fm_tbl.c, 1135
fm_tbl.h, 1140
FM_ValidateTable
fm_tbl.c, 1136
fm_tbl.h, 1141
FM_VerifyChildTask
fm_cmd_utils.c, 1035
fm_cmd_utils.h, 1054
FM_VerifyDirExists
fm_cmd_utils.c, 1036
fm_cmd_utils.h, 1056
FM_VerifyDirNoExist
fm_cmd_utils.c, 1037
fm_cmd_utils.h, 1057
FM_VerifyFileClosed
fm_cmd_utils.c, 1039
fm_cmd_utils.h, 1058
FM_VerifyFileExists
fm_cmd_utils.c, 1040
fm_cmd_utils.h, 1060
FM_VerifyFileNoExist
fm_cmd_utils.c, 1041
fm_cmd_utils.h, 1061
FM_VerifyFileNotOpen
fm_cmd_utils.c, 1042
fm_cmd_utils.h, 1062
FM_VerifyNameValid
fm_cmd_utils.c, 1044
fm_cmd_utils.h, 1063
FM_VerifyOverwrite
fm_cmd_utils.c, 1045
fm_cmd_utils.h, 1065
FailedValCounter
CFE_TBL_HousekeepingTlm_Payload, 814
FileCreateTimeSecs
CFE_TBL_Info, 820
CFE_TBL_TblRegPacket_Payload, 832
FileCreateTimeSubSecs
CFE_TBL_Info, 820
CFE_TBL_TblRegPacket_Payload, 832
FileEntries
FM_DirListFileStats_t, 885
FileInfoCRC
FM_ChildQueueEntry_t, 872
FM_GetFileInfoCmd_t, 898
FileInfoPkt
FM_GlobalData_t, 907
FileInfoSize
FM_ChildQueueEntry_t, 872
FileInfoState
FM_ChildQueueEntry_t, 872
FileInfoTime
FM_ChildQueueEntry_t, 873
FileIsOpen
fm_cmd_utils.c, 1048
FileList
FM_DirListPkt_t, 887
FileModeBits

os_fstat_t, 932
FileName
CFE_ES_AppInfo, 670
CFE_ES_FileNameCmd_Payload, 683
CFE_FS_FileWriteMetaData, 760
FM_SetPermCmd_t, 922
os_dirent_t, 927
FileSize
FM_FileInfoPkt_t, 889
os_fstat_t, 932
FileStatMode
FM_GlobalData_t, 907
FileStatSize
FM_GlobalData_t, 907
FileStatTime
FM_GlobalData_t, 908
FileStatus
FM_FileInfoPkt_t, 890
FileSubType
CFE_FS_FileWriteMetaData, 761
FileSys
FM_FreeSpacePkt_t, 891
FM_FreeSpaceTable_t, 894
FileTime
os_fstat_t, 932
Filename
CFE_SB_WriteFileInfoCmd_Payload, 798
FM_DeleteFileCmd_t, 882
FM_FileInfoPkt_t, 889
FM_GetDirFileCmd_t, 895
FM_GetFileInfoCmd_t, 899
filename
OS_module_prop_t, 936
FilterMask
CFE_ES_SetPerfFilterMaskCmd_Payload, 715
FilterMaskNum
CFE_ES_SetPerfFilterMaskCmd_Payload, 715
FirstFile
FM_DirListPkt_t, 887
flags
OS_module_address_t, 935
fm_app.c
FM_AppInit, 952
FM_AppMain, 953
FM_GlobalData, 962
FM_ProcessCmd, 955
FM_ProcessPkt, 958
FM_ReportHK, 960
fm_app.h
FM_AppInit, 963
FM_AppMain, 964
FM_ProcessCmd, 966
FM_ProcessPkt, 968
FM_ReportHK, 970
FM_SB_TIMEOUT, 963
fm_child.c
FM_ChildConcatCmd, 974
FM_ChildCopyCmd, 975
FM_ChildCreateDirCmd, 976
FM_ChildDeleteAllCmd, 977
FM_ChildDeleteCmd, 978
FM_ChildDeleteDirCmd, 979
FM_ChildDirListFileCmd, 981
FM_ChildDirListFileInit, 982
FM_ChildDirListFileLoop, 984
FM_ChildDirListPktCmd, 985
FM_ChildFileInfoCmd, 986
FM_ChildInit, 988
FM_ChildLoop, 990
FM_ChildMoveCmd, 991
FM_ChildProcess, 992
FM_ChildRenameCmd, 994
FM_ChildSetPermissionsCmd, 995
FM_ChildSizeTimeMode, 996
FM_ChildSleepStat, 997
FM_ChildTask, 998
FM_QUEUE_SEM_NAME, 973
OS_DIRENTRY_NAME, 973
fm_child.h
FM_ChildConcatCmd, 1001
FM_ChildCopyCmd, 1003
FM_ChildCreateDirCmd, 1004
FM_ChildDeleteAllCmd, 1005
FM_ChildDeleteCmd, 1006
FM_ChildDeleteDirCmd, 1007
FM_ChildDirListFileCmd, 1009
FM_ChildDirListFileInit, 1010
FM_ChildDirListFileLoop, 1012
FM_ChildDirListPktCmd, 1013
FM_ChildFileInfoCmd, 1014
FM_ChildInit, 1016
FM_ChildLoop, 1018
FM_ChildMoveCmd, 1019
FM_ChildProcess, 1020
FM_ChildRenameCmd, 1022
FM_ChildSetPermissionsCmd, 1023
FM_ChildSizeTimeMode, 1024
FM_ChildSleepStat, 1025
FM_ChildTask, 1026
fm_cmd_utils.c
FM_AppendPathSep, 1029
FM_GetFilenameState, 1030
FM_GetOpenFilesData, 1031
FM_InvokeChildTask, 1032
FM_IsValidCmdPktLength, 1033
FM_VerifyChildTask, 1035
FM_VerifyDirExists, 1036
FM_VerifyDirNoExist, 1037

FM_VerifyFileClosed, 1039
FM_VerifyFileExists, 1040
FM_VerifyFileNoExist, 1041
FM_VerifyFileNotOpen, 1042
FM_VerifyNameValid, 1044
FM_VerifyOverwrite, 1045
FileIsOpen, 1048
LoadOpenFileData, 1046
OpenFileCount, 1048
SearchOpenFileData, 1047
fm_cmd_utils.h
FM_AppendPathSep, 1049
FM_GetFilenameState, 1050
FM_GetOpenFilesData, 1051
FM_InvokeChildTask, 1052
FM_IsValidCmdPktLength, 1053
FM_VerifyChildTask, 1054
FM_VerifyDirExists, 1056
FM_VerifyDirNoExist, 1057
FM_VerifyFileClosed, 1058
FM_VerifyFileExists, 1060
FM_VerifyFileNoExist, 1061
FM_VerifyFileNotOpen, 1062
FM_VerifyNameValid, 1063
FM_VerifyOverwrite, 1065
fm_cmds.c
FM_ConcatFilesCmd, 1067
FM_CopyFileCmd, 1069
FM_CreateDirectoryCmd, 1070
FM_DeleteAllFilesCmd, 1071
FM_DeleteDirectoryCmd, 1073
FM_DeleteFileCmd, 1074
FM_GetDirListFileCmd, 1075
FM_GetDirListPktCmd, 1077
FM_GetFileInfoCmd, 1078
FM_GetFreeSpaceCmd, 1080
FM_GetOpenFilesCmd, 1081
FM_MoveFileCmd, 1082
FM_NoopCmd, 1084
FM_RenameFileCmd, 1085
FM_ResetCountersCmd, 1086
FM_SetPermissionsCmd, 1087
FM_SetTableStateCmd, 1088
fm_cmds.h
FM_ConcatFilesCmd, 1091
FM_CopyFileCmd, 1092
FM_CreateDirectoryCmd, 1094
FM_DeleteAllFilesCmd, 1096
FM_DeleteDirectoryCmd, 1097
FM_DeleteFileCmd, 1098
FM_GetDirListFileCmd, 1100
FM_GetDirListPktCmd, 1101
FM_GetFileInfoCmd, 1103
FM_GetFreeSpaceCmd, 1104
FM_GetOpenFilesCmd, 1105
FM_MoveFileCmd, 1107
FM_NoopCmd, 1108
FM_RenameFileCmd, 1109
FM_ResetCountersCmd, 1110
FM_SetPermissionsCmd, 1112
FM_SetTableStateCmd, 1113
fm_defs.h
FM_IGNORE_CRC, 1115
FM_NAME_IS_DIRECTORY, 1115
FM_NAME_IS_FILE_CLOSED, 1115
FM_NAME_IS_FILE_OPEN, 1115
FM_NAME_IS_INVALID, 1115
FM_NAME_IS_NOT_IN_USE, 1116
FM_PARENT_DIRECTORY, 1116
FM_TABLE_ENTRY_DISABLED, 1116
FM_TABLE_ENTRY_ENABLED, 1116
FM_TABLE_ENTRY_UNUSED, 1116
FM_THIS_DIRECTORY, 1117
fm_freespace.c
__attribute__, 1144
FM_FreeSpaceTable, 1144
fm_msg.h
FM_GlobalData, 1131
fm_tbl.c
FM_AcquireTablePointers, 1133
FM_ReleaseTablePointers, 1134
FM_TableInit, 1135
FM_ValidateTable, 1136
fm_tbl.h
FM_AcquireTablePointers, 1138
FM_ReleaseTablePointers, 1139
FM_TableInit, 1140
FM_ValidateTable, 1141
Forced2Fly
CFE_TIME_DiagnosticTlm_Payload, 844
free_blocks
OS_heap_prop_t, 933
free_bytes
OS_heap_prop_t, 933
FreeFds
os_fsinfo_t, 930
FreeSpace
FM_FreeSpacePktEntry_t, 893
FreeSpacePkt
FM_GlobalData_t, 908
FreeSpaceTableHandle
FM_GlobalData_t, 908
FreeSpaceTablePtr
FM_GlobalData_t, 908
FreeVolumes
os_fsinfo_t, 930
freerun_time
OS_timebase_prop_t, 947

GetData
CFE_FS_FileWriteMetaData, 761
GetPipeIdByNameErrorCounter
CFE_SB_HousekeepingTlm_Payload, 770
GetSizeTimeMode
FM_ChildQueueEntry_t, 873
FM_GetDirFileCmd_t, 895
FM_GetDirPktCmd_t, 897

Handle
CFE_ES_CDSRegDumpRec, 677
HeapBlocksFree
CFE_ES_HousekeepingTlm_Payload, 689
HeapBytesFree
CFE_ES_HousekeepingTlm_Payload, 689
HeapMaxBlockSize
CFE_ES_HousekeepingTlm_Payload, 689
host_module_id
OS_module_prop_t, 936
HousekeepingPkt
FM_GlobalData_t, 909

InactiveBufferAddr
CFE_TBL_TblRegPacket_Payload, 832
Index
CFE_SB_MsgMapFileEntry, 776
int16
common_types.h, 1597
int32
common_types.h, 1597
int64
common_types.h, 1597
int8
common_types.h, 1597
InternalErrorCounter
CFE_SB_HousekeepingTlm_Payload, 770
interval_time
OS_timer_prop_t, 948
intptr
common_types.h, 1598
IsPending
CFE_FS_FileWriteMetaData, 761
IsValid
OS_file_prop_t, 929

LENGTHCHECK
osapi-macros.h, 1625
largest_free_block
OS_heap_prop_t, 933
LastFileDumped
CFE_TBL_HousekeepingTlm_Payload, 814
LastFileLoaded
CFE_TBL_HousekeepingTlm_Payload, 815
CFE_TBL_Info, 820
CFE_TBL_TblRegPacket_Payload, 832
LastModifiedTime
FM_FileInfoPkt_t, 890
LastTableLoaded
CFE_TBL_HousekeepingTlm_Payload, 815
LastUpdateTime
CFE_TBL_HousekeepingTlm_Payload, 815
LastUpdatedTable
CFE_TBL_HousekeepingTlm_Payload, 815
LastValCrc
CFE_TBL_HousekeepingTlm_Payload, 816
LastValStatus
CFE_TBL_HousekeepingTlm_Payload, 816
LastValTableName
CFE_TBL_HousekeepingTlm_Payload, 816
LeapSeconds
CFE_TIME_HousekeepingTlm_Payload, 853
CFE_TIME_LeapsCmd_Payload, 856
Length
CCSDS_PrimaryHeader, 665
CFE_FS_Header, 763
LoadFilename
CFE_TBL_LoadCmd_Payload, 823
LoadOpenFileData
fm_cmd_utils.c, 1046
LoadPending
CFE_TBL_TblRegPacket_Payload, 833
LocalIntCounter
CFE_TIME_DiagnosticTlm_Payload, 844
LocalTaskCounter
CFE_TIME_DiagnosticTlm_Payload, 844
LogEnabled
CFE_EVS_HousekeepingTlm_Payload, 743
LogFilename
CFE_EVS_LogFileCmd_Payload, 747
LogFullFlag
CFE_EVS_HousekeepingTlm_Payload, 743
LogMode
CFE_EVS_HousekeepingTlm_Payload, 744
CFE_EVS_SetLogMode_Payload, 755
LogOverflowCounter
CFE_EVS_HousekeepingTlm_Payload, 744
LogicalName
FM_OpenFilesEntry_t, 917
LongDouble
CFE_ES_PoolAlign, 705
CFE_SB_Msg, 774
LongInt
CFE_ES_PoolAlign, 705
CFE_SB_Msg, 774

MainTaskId
CFE_ES_AppInfo, 670
MainTaskName
CFE_ES_AppInfo, 670

Mask
CFE_EVS_AppNameEventIDMaskCmd_Payload, 735
CFE_EVS_BinFilter, 738
MaxElapsed
CFE_TIME_DiagnosticTlm_Payload, 844
MaxFds
os_fsinfo_t, 930
MaxLocalClock
CFE_TIME_DiagnosticTlm_Payload, 845
MaxMemAllowed
CFE_SB_StatsTlm_Payload, 791
MaxMsgIdsAllowed
CFE_SB_StatsTlm_Payload, 791
MaxPRCount
CFE_ES_SetMaxPRCountCmd_Payload, 713
MaxPipeDepthAllowed
CFE_SB_StatsTlm_Payload, 791
MaxPipesAllowed
CFE_SB_StatsTlm_Payload, 792
MaxProcessorResets
CFE_ES_HousekeepingTlm_Payload, 690
MaxQueueDepth
CFE_SB_PipeDepthStats, 777
CFE_SB_PipeInfoEntry, 780
MaxSubscriptionsAllowed
CFE_SB_StatsTlm_Payload, 792
MaxVolumes
os_fsinfo_t, 931
MemInUse
CFE_SB_HousekeepingTlm_Payload, 770
CFE_SB_StatsTlm_Payload, 792
MemPoolHandle
CFE_SB_HousekeepingTlm_Payload, 770
CFE_TBL_HousekeepingTlm_Payload, 816
Message
CFE_EVS_LongEventTlm_Payload, 749
MessageFormatMode
CFE_EVS_HousekeepingTlm_Payload, 744
MessageSendCounter
CFE_EVS_HousekeepingTlm_Payload, 744
MessageTruncCounter
CFE_EVS_HousekeepingTlm_Payload, 745
MicroSeconds
CFE_TIME_TimeCmd_Payload, 867
MinElapsed
CFE_TIME_DiagnosticTlm_Payload, 845
Mode
CFE_ES_OverWriteSysLogCmd_Payload, 704
FM_ChildQueueEntry_t, 873
FM_DirListEntry_t, 884
FM_FileInfoPkt_t, 890
FM_SetPermCmd_t, 922
ModifyTime
FM_DirListEntry_t, 884
Module
OS_static_symbol_record_t, 942
Msg
CFE_SB_Msg, 774
MsgCnt
CFE_SB_RoutingFileEntry, 785
MsgFormat
CFE_EVS_SetEventFormatCode_Payload, 753
MsgId
CFE_SB_MsgMapFileEntry, 776
CFE_SB_RouteCmd_Payload, 784
CFE_SB_RoutingFileEntry, 786
CFE_SB_SingleSubscriptionTlm_Payload, 788
CFE_SB_SubEntries, 796
MsgIdsInUse
CFE_SB_StatsTlm_Payload, 792
MsgLimitErrorCounter
CFE_SB_HousekeepingTlm_Payload, 771
MsgReceiveErrorCounter
CFE_SB_HousekeepingTlm_Payload, 771
MsgSendErrorCounter
CFE_SB_HousekeepingTlm_Payload, 771

Name
CFE_ES_AppInfo, 670
CFE_ES_CDSRegDumpRec, 677
CFE_TBL_TblRegPacket_Payload, 833
FM_FreeSpacePktEntry_t, 893
FM_TableEntry_t, 924
OS_static_symbol_record_t, 943
name
OS_bin_sem_prop_t, 925
OS_count_sem_prop_t, 926
OS_module_prop_t, 936
OS_mut_sem_prop_t, 937
OS_queue_prop_t, 938
OS_socket_prop_t, 941
OS_task_prop_t, 945
OS_timebase_prop_t, 947
OS_timer_prop_t, 948
NoSubscribersCounter
CFE_SB_HousekeepingTlm_Payload, 771
nominal_interval_time
OS_timebase_prop_t, 947
NumBlocksRequested
CFE_ES_MemPoolStats, 698
NumBytes
CFE_TBL_File_Hdr, 809
NumCreated
CFE_ES_BlockStats, 676
NumFree
CFE_ES_BlockStats, 676
NumFreeBytes

CFE_ES_MemPoolStats, 699
NumFreeSharedBufs
CFE_TBL_HousekeepingTlm_Payload, 817
NumLoadPending
CFE_TBL_HousekeepingTlm_Payload, 817
NumOfChildTasks
CFE_ES_AppInfo, 671
NumOpenFiles
FM_HousekeepingPkt_t, 913
FM_OpenFilesPkt_t, 918
NumTables
CFE_TBL_HousekeepingTlm_Payload, 817
NumUsers
CFE_TBL_Info, 821
NumValRequests
CFE_TBL_HousekeepingTlm_Payload, 817

OS_API_Init
OSAL Core Operation APIs, 543
OS_API_Teardown
OSAL Core Operation APIs, 544
OS_Application_Run
OSAL Core Operation APIs, 544
OS_Application_Startup
OSAL Core Operation APIs, 544
OS_ApplicationExit
OSAL Core Operation APIs, 544
OS_ApplicationShutdown
OSAL Core Operation APIs, 545
OS_ArgCallback_t
common_types.h, 1598
OS_BSP_GetArgC
OSAL BSP low level access APIs, 530
OS_BSP_GetArgV
OSAL BSP low level access APIs, 530
OS_BSP_GetResourceTypeConfig
OSAL BSP low level access APIs, 530
OS_BSP_SetExitCode
OSAL BSP low level access APIs, 530
OS_BSP_SetResourceTypeConfig
OSAL BSP low level access APIs, 530
OS_BUFFER_MSG_DEPTH
osconfig-example.h, 1145
OS_BUFFER_SIZE
osconfig-example.h, 1145
OS_BUILD_BASELINE
osapi-version.h, 1641
OS_BUILD_NUMBER
osapi-version.h, 1641
OS_BinSemCreate
OSAL Binary Semaphore APIs, 524
OS_BinSemDelete
OSAL Binary Semaphore APIs, 525
OS_BinSemFlush
OSAL Binary Semaphore APIs, 525
OS_BinSemGetIdByName
OSAL Binary Semaphore APIs, 526
OS_BinSemGetInfo
OSAL Binary Semaphore APIs, 527
OS_BinSemGive
OSAL Binary Semaphore APIs, 527
OS_BinSemTake
OSAL Binary Semaphore APIs, 528
OS_BinSemTimedWait
OSAL Binary Semaphore APIs, 528
OS_CHECK
osapi-constants.h, 1608
OS_CHK_ONLY
osapi-filesys.h, 1620
OS_CloseAllFiles
OSAL Standard File APIs, 575
OS_CloseFileByName
OSAL Standard File APIs, 575
OS_ConvertToArrayIndex
OSAL Object ID Utility APIs, 600
OS_CountSemCreate
OSAL Counting Semaphore APIs, 547
OS_CountSemDelete
OSAL Counting Semaphore APIs, 548
OS_CountSemGetIdByName
OSAL Counting Semaphore APIs, 549
OS_CountSemGetInfo
OSAL Counting Semaphore APIs, 549
OS_CountSemGive
OSAL Counting Semaphore APIs, 550
OS_CountSemTake
OSAL Counting Semaphore APIs, 550
OS_CountSemTimedWait
OSAL Counting Semaphore APIs, 551
OS_DIRENTRY_NAME
fm_child.c, 973
osapi-dir.h, 1611
OS_DeleteAllObjects
OSAL Core Operation APIs, 545
OS_DirectoryClose
OSAL Directory APIs, 553
OS_DirectoryOpen
OSAL Directory APIs, 554
OS_DirectoryRead
OSAL Directory APIs, 554
OS_DirectoryRewind
OSAL Directory APIs, 555
OS_ERR_BAD_ADDRESS
OSAL Return Code Defines, 560
OS_ERR_FILE
OSAL Return Code Defines, 560
OS_ERR_INCORRECT_OBJ_STATE
OSAL Return Code Defines, 560

OS_ERR_INCORRECT_OBJ_TYPE
OSAL Return Code Defines, 561
OS_ERR_INVALID_ARGUMENT
OSAL Return Code Defines, 561
OS_ERR_INVALID_ID
OSAL Return Code Defines, 561
OS_ERR_INVALID_PRIORITY
OSAL Return Code Defines, 561
OS_ERR_INVALID_SIZE
OSAL Return Code Defines, 561
OS_ERR_NAME_NOT_FOUND
OSAL Return Code Defines, 562
OS_ERR_NAME_TAKEN
OSAL Return Code Defines, 562
OS_ERR_NAME_TOO_LONG
OSAL Return Code Defines, 562
OS_ERR_NO_FREE_IDS
OSAL Return Code Defines, 562
OS_ERR_NOT_IMPLEMENTED
OSAL Return Code Defines, 562
OS_ERR_OBJECT_IN_USE
OSAL Return Code Defines, 563
OS_ERR_OPERATION_NOT_SUPPORTED
OSAL Return Code Defines, 563
OS_ERR_OUTPUT_TOO_LARGE
OSAL Return Code Defines, 563
OS_ERR_SEM_NOT_FULL
OSAL Return Code Defines, 563
OS_ERR_STREAM_DISCONNECTED
OSAL Return Code Defines, 563
OS_ERROR_ADDRESS_MISALIGNED
OSAL Return Code Defines, 564
OS_ERROR_NAME_LENGTH
osapi-error.h, 1614
OS_ERROR_TIMEOUT
OSAL Return Code Defines, 564
OS_ERROR
OSAL Return Code Defines, 564
OS_Event_t
osapi-common.h, 1607
OS_EventHandler_t
osapi-common.h, 1606
OS_FDGetInfo
OSAL Standard File APIs, 576
OS_FILESTAT_EXEC
osapi-file.h, 1616
OS_FILESTAT_ISDIR
osapi-file.h, 1617
OS_FILESTAT_MODE
osapi-file.h, 1617
OS_FILESTAT_READ
osapi-file.h, 1617
OS_FILESTAT_SIZE
osapi-file.h, 1617
OS_FILESTAT_TIME
osapi-file.h, 1618
OS_FILESTAT_WRITE
osapi-file.h, 1618
OS_FP_ENABLED
osapi-task.h, 1636
OS_FS_DEV_NAME_LEN
osconfig-example.h, 1146
OS_FS_ERR_DEVICE_NOT_FREE
OSAL Return Code Defines, 564
OS_FS_ERR_DRIVE_NOT_CREATED
OSAL Return Code Defines, 564
OS_FS_ERR_NAME_TOO_LONG
OSAL Return Code Defines, 565
OS_FS_ERR_PATH_INVALID
OSAL Return Code Defines, 565
OS_FS_ERR_PATH_TOO_LONG
OSAL Return Code Defines, 565
OS_FS_GetPhysDriveName
OSAL File System Level APIs, 588
OS_FS_PHYS_NAME_LEN
osconfig-example.h, 1146
OS_FS_VOL_NAME_LEN
osconfig-example.h, 1146
OS_FdSet, 928
object_ids, 928
OS_FileOpenCheck
OSAL Standard File APIs, 577
OS_FileSysAddFixedMap
OSAL File System Level APIs, 587
OS_FileSysStatVolume
OSAL File System Level APIs, 588
OS_ForEachObject
OSAL Object ID Utility APIs, 601
OS_ForEachObjectOfType
OSAL Object ID Utility APIs, 601
OS_GetBuildNumber
osapi-version.h, 1644
OS_GetErrorName
OSAL Error Info APIs, 569
OS_GetFsInfo
OSAL File System Level APIs, 589
OS_GetLocalTime
OSAL Real Time Clock APIs, 531
OS_GetResourceName
OSAL Object ID Utility APIs, 602
OS_GetVersionCodeName
osapi-version.h, 1644
OS_GetVersionNumber
osapi-version.h, 1644
OS_GetVersionString
osapi-version.h, 1645
OS_HeapGetInfo
OSAL Heap APIs, 595

OS_INVALID_INT_NUM
OSAL Return Code Defines, 565
OS_INVALID_POINTER
OSAL Return Code Defines, 565
OS_INVALID_SEM_VALUE
OSAL Return Code Defines, 566
OS_IdentifyObject
OSAL Object ID Utility APIs, 603
OS_IdleLoop
OSAL Core Operation APIs, 545
OS_MAJOR_VERSION
osapi-version.h, 1641
OS_MAX_API_NAME
osconfig-example.h, 1146
OS_MAX_BIN_SEMAPHORES
osconfig-example.h, 1146
OS_MAX_CMD_LEN
osconfig-example.h, 1147
OS_MAX_CONSOLES
osconfig-example.h, 1147
OS_MAX_COUNT_SEMAPHORES
osconfig-example.h, 1147
OS_MAX_FILE_NAME
osconfig-example.h, 1147
OS_MAX_FILE_SYSTEMS
osconfig-example.h, 1148
OS_MAX_LOCAL_PATH_LEN
osapi-constants.h, 1608
OS_MAX_MODULES
osconfig-example.h, 1148
OS_MAX_MUTEXES
osconfig-example.h, 1148
OS_MAX_NUM_OPEN_DIRS
osconfig-example.h, 1148
OS_MAX_NUM_OPEN_FILES
osconfig-example.h, 1149
OS_MAX_PATH_LEN
osconfig-example.h, 1149
OS_MAX_QUEUES
osconfig-example.h, 1149
OS_MAX_SYM_LEN
osconfig-example.h, 1150
OS_MAX_TASK_PRIORITY
osapi-task.h, 1636
OS_MAX_TASKS
osconfig-example.h, 1150
OS_MAX_TIMEBASES
osconfig-example.h, 1150
OS_MAX_TIMERS
osconfig-example.h, 1151
OS_MINOR_VERSION
osapi-version.h, 1641
OS_MISSION_REV
osapi-version.h, 1642
OS_MODULE_FILE_EXTENSION
osconfig-example.h, 1151
OS_MODULE_FLAG_GLOBAL_SYMBOLS
osapi-module.h, 1626
OS_MODULE_FLAG_LOCAL_SYMBOLS
osapi-module.h, 1627
OS_ModuleInfo
OSAL Dynamic Loader and Symbol APIs, 607
OS_ModuleLoad
OSAL Dynamic Loader and Symbol APIs, 608
OS_ModuleSymbolLookup
OSAL Dynamic Loader and Symbol APIs, 608
OS_ModuleUnload
OSAL Dynamic Loader and Symbol APIs, 609
OS_MutSemCreate
OSAL Mutex APIs, 612
OS_MutSemDelete
OSAL Mutex APIs, 613
OS_MutSemGetIdByName
OSAL Mutex APIs, 613
OS_MutSemGetInfo
OSAL Mutex APIs, 614
OS_MutSemGive
OSAL Mutex APIs, 614
OS_MutSemTake
OSAL Mutex APIs, 615
OS_NetworkGetHostName
OSAL Network ID APIs, 617
OS_NetworkGetID
OSAL Network ID APIs, 617
OS_OBJECT_CREATOR_ANY
osapi-constants.h, 1608
OS_OBJECT_ID_UNDEFINED
osapi-constants.h, 1609
OS_OBJECT_INDEX_MASK
osapi-idmap.h, 1623
OS_OBJECT_TYPE_OS_BINSEM
OSAL Object Type Defines, 596
OS_OBJECT_TYPE_OS_CONSOLE
OSAL Object Type Defines, 596
OS_OBJECT_TYPE_OS_COUNTSEM
OSAL Object Type Defines, 597
OS_OBJECT_TYPE_OS_DIR
OSAL Object Type Defines, 597
OS_OBJECT_TYPE_OS_FILESYS
OSAL Object Type Defines, 597
OS_OBJECT_TYPE_OS_MODULE
OSAL Object Type Defines, 597
OS_OBJECT_TYPE_OS_MUTEX
OSAL Object Type Defines, 597
OS_OBJECT_TYPE_OS_QUEUE
OSAL Object Type Defines, 598
OS_OBJECT_TYPE_OS_STREAM
OSAL Object Type Defines, 598

OS_OBJECT_TYPE_OS_TASK
OSAL Object Type Defines, 598
OS_OBJECT_TYPE_OS_TIMEBASE
OSAL Object Type Defines, 598
OS_OBJECT_TYPE_OS_TIMECB
OSAL Object Type Defines, 598
OS_OBJECT_TYPE_SHIFT
osapi-idmap.h, 1623
OS_OBJECT_TYPE_UNDEFINED
OSAL Object Type Defines, 599
OS_OBJECT_TYPE_USER
OSAL Object Type Defines, 599
OS_ObjectIdDefined
OSAL Object ID Utility APIs, 603
OS_ObjectIdEqual
OSAL Object ID Utility APIs, 604
OS_ObjectIdFromInteger
OSAL Object ID Utility APIs, 604
OS_ObjectIdToArrayIndex
OSAL Object ID Utility APIs, 605
OS_ObjectIdToInteger
OSAL Object ID Utility APIs, 605
OS_OpenCreate
OSAL Standard File APIs, 579
OS_PEND
osapi-constants.h, 1609
OS_PRINTF_CONSOLE_NAME
osconfig-example.h, 1151
OS_PRINTF
cfe_es.h, 1262
common_types.h, 1595
OS_QUEUE_EMPTY
OSAL Return Code Defines, 566
OS_QUEUE_FULL
OSAL Return Code Defines, 566
OS_QUEUE_ID_ERROR
OSAL Return Code Defines, 566
OS_QUEUE_INVALID_SIZE
OSAL Return Code Defines, 566
OS_QUEUE_MAX_DEPTH
osconfig-example.h, 1151
OS_QUEUE_TIMEOUT
OSAL Return Code Defines, 567
OS_QueueCreate
OSAL Message Queue APIs, 621
OS_QueueDelete
OSAL Message Queue APIs, 622
OS_QueueGet
OSAL Message Queue APIs, 623
OS_QueueGetIdByName
OSAL Message Queue APIs, 623
OS_QueueGetInfo
OSAL Message Queue APIs, 624
OS_QueuePut
OSAL Message Queue APIs, 624
OS_READ_ONLY
OSAL File Access Option Defines, 571
OS_READ_WRITE
OSAL File Access Option Defines, 571
OS_REPAIR
osapi-filesys.h, 1620
OS_REVISION
osapi-version.h, 1642
OS_RegisterEventHandler
OSAL Core Operation APIs, 546
OS_SEEK_CUR
OSAL Reference Point For Seek Offset Defines, 572
OS_SEEK_END
OSAL Reference Point For Seek Offset Defines, 572
OS_SEEK_SET
OSAL Reference Point For Seek Offset Defines, 572
OS_SEM_EMPTY
OSAL Semaphore State Defines, 523
OS_SEM_FAILURE
OSAL Return Code Defines, 567
OS_SEM_FULL
OSAL Semaphore State Defines, 523
OS_SEM_TIMEOUT
OSAL Return Code Defines, 567
OS_SHELL_CMD_INPUT_FILE_NAME
osconfig-example.h, 1152
OS_SOCKADDR_MAX_LEN
osapi-sockets.h, 1633
osconfig-example.h, 1152
OS_STR_HELPER
osapi-version.h, 1642
OS_STR
osapi-version.h, 1642
OS_SUCCESS
OSAL Return Code Defines, 567
OS_SelectFdAdd
OSAL Select APIs, 626
OS_SelectFdClear
OSAL Select APIs, 627
OS_SelectFdIsSet
OSAL Select APIs, 627
OS_SelectFdZero
OSAL Select APIs, 628
OS_SelectMultiple
OSAL Select APIs, 628
OS_SelectSingle
OSAL Select APIs, 629
OS_SetLocalTime
OSAL Real Time Clock APIs, 532
OS_ShellOutputToFile
OSAL Shell APIs, 631
OS_SockAddr_t, 938
ActualLength, 939

AddrData, 939
OS_SockAddrData_t, 939
AlignPtr, 940
AlignU32, 940
Buffer, 940
OS_SocketAccept
OSAL Socket Management APIs, 636
OS_SocketAddrFromString
OSAL Socket Address APIs, 632
OS_SocketAddrGetPort
OSAL Socket Address APIs, 633
OS_SocketAddrInit
OSAL Socket Address APIs, 633
OS_SocketAddrSetPort
OSAL Socket Address APIs, 634
OS_SocketAddrToString
OSAL Socket Address APIs, 635
OS_SocketBind
OSAL Socket Management APIs, 638
OS_SocketConnect
OSAL Socket Management APIs, 639
OS_SocketDomain_t
osapi-sockets.h, 1633
OS_SocketGetIdByName
OSAL Socket Management APIs, 639
OS_SocketGetInfo
OSAL Socket Management APIs, 640
OS_SocketOpen
OSAL Socket Management APIs, 641
OS_SocketRecvFrom
OSAL Socket Management APIs, 641
OS_SocketSendTo
OSAL Socket Management APIs, 642
OS_SocketShutdown
OSAL Socket Management APIs, 643
OS_SocketShutdownMode_t
osapi-sockets.h, 1634
OS_SocketType_t
osapi-sockets.h, 1634
OS_StatusToInteger
OSAL Error Info APIs, 569
OS_StreamState_t
osapi-select.h, 1630
OS_SymbolLookup
OSAL Dynamic Loader and Symbol APIs, 610
OS_SymbolTableDump
OSAL Dynamic Loader and Symbol APIs, 610
OS_TIMER_ERR_INTERNAL
OSAL Return Code Defines, 567
OS_TIMER_ERR_INVALID_ARGS
OSAL Return Code Defines, 568
OS_TIMER_ERR_TIMER_ID
OSAL Return Code Defines, 568
OS_TIMER_ERR_UNAVAILABLE
OSAL Return Code Defines, 568
OS_TaskCreate
OSAL Task APIs, 644
OS_TaskDelay
OSAL Task APIs, 645
OS_TaskDelete
OSAL Task APIs, 646
OS_TaskExit
OSAL Task APIs, 646
OS_TaskFindIdBySystemData
OSAL Task APIs, 646
OS_TaskGetId
OSAL Task APIs, 647
OS_TaskGetIdByName
OSAL Task APIs, 647
OS_TaskGetInfo
OSAL Task APIs, 648
OS_TaskInstallDeleteHandler
OSAL Task APIs, 649
OS_TaskSetPriority
OSAL Task APIs, 649
OS_TimeAdd
OSAL Real Time Clock APIs, 533
OS_TimeAssembleFromMicroseconds
OSAL Real Time Clock APIs, 533
OS_TimeAssembleFromMilliseconds
OSAL Real Time Clock APIs, 534
OS_TimeAssembleFromNanoseconds
OSAL Real Time Clock APIs, 534
OS_TimeAssembleFromSubseconds
OSAL Real Time Clock APIs, 535
OS_TimeBaseCreate
OSAL Time Base APIs, 651
OS_TimeBaseDelete
OSAL Time Base APIs, 652
OS_TimeBaseGetFreeRun
OSAL Time Base APIs, 653
OS_TimeBaseGetIdByName
OSAL Time Base APIs, 654
OS_TimeBaseGetInfo
OSAL Time Base APIs, 654
OS_TimeBaseSet
OSAL Time Base APIs, 655
OS_TimeGetFractionalPart
OSAL Real Time Clock APIs, 536
OS_TimeGetMicrosecondsPart
OSAL Real Time Clock APIs, 536
OS_TimeGetMillisecondsPart
OSAL Real Time Clock APIs, 537
OS_TimeGetNanosecondsPart
OSAL Real Time Clock APIs, 538
OS_TimeGetSubsecondsPart
OSAL Real Time Clock APIs, 539
OS_TimeGetTotalMicroseconds

OSAL Real Time Clock APIs, 539
OS_TimeGetTotalMilliseconds
OSAL Real Time Clock APIs, 540
OS_TimeGetTotalNanoseconds
OSAL Real Time Clock APIs, 540
OS_TimeGetTotalSeconds
OSAL Real Time Clock APIs, 541
OS_TimeSubtract
OSAL Real Time Clock APIs, 542
OS_TimedRead
OSAL Standard File APIs, 583
OS_TimedWrite
OSAL Standard File APIs, 584
OS_TimerAdd
OSAL Timer APIs, 657
OS_TimerCallback_t
osapi-timer.h, 1640
OS_TimerCreate
OSAL Timer APIs, 659
OS_TimerDelete
OSAL Timer APIs, 660
OS_TimerGetIdByName
OSAL Timer APIs, 661
OS_TimerGetInfo
OSAL Timer APIs, 662
OS_TimerSet
OSAL Timer APIs, 663
OS_TimerSync_t
osapi-timebase.h, 1639
OS_TranslatePath
OSAL File System Level APIs, 593
OS_USED
common_types.h, 1595
OS_UTILITYTASK_PRIORITY
osconfig-example.h, 1152
OS_UTILITYTASK_STACK_SIZE
osconfig-example.h, 1152
OS_VERSION_CODENAME
osapi-version.h, 1643
OS_VERSION_STRING
osapi-version.h, 1643
OS_VERSION
osapi-version.h, 1642
OS_WRITE_ONLY
OSAL File Access Option Defines, 571
OS_bin_sem_prop_t, 925
creator, 925
name, 925
value, 926
OS_chkfs
OSAL File System Level APIs, 586
OS_chmod
OSAL Standard File APIs, 573
OS_close
OSAL Standard File APIs, 574
OS_count_sem_prop_t, 926
creator, 926
name, 926
value, 927
OS_cp
OSAL Standard File APIs, 576
OS_file_flag_t
osapi-file.h, 1619
OS_file_prop_t, 928
IsValid, 929
Path, 929
User, 929
OS_heap_prop_t, 932
free_blocks, 933
free_bytes, 933
largest_free_block, 933
OS_initfs
OSAL File System Level APIs, 590
OS_lseek
OSAL Standard File APIs, 577
OS_mkdir
OSAL Directory APIs, 556
OS_mkfs
OSAL File System Level APIs, 591
OS_module_address_t, 933
bss_address, 934
bss_size, 934
code_address, 934
code_size, 934
data_address, 935
data_size, 935
flags, 935
valid, 935
OS_module_prop_t, 935
addr, 936
entry_point, 936
filename, 936
host_module_id, 936
name, 936
OS_mount
OSAL File System Level APIs, 591
OS_mut_sem_prop_t, 937
creator, 937
name, 937
OS_mv
OSAL Standard File APIs, 578
OS_printf
OSAL Printf APIs, 619
OS_printf_disable
OSAL Printf APIs, 619
OS_printf_enable
OSAL Printf APIs, 620
OS_queue_prop_t, 938

creator, 938
name, 938
OS_read
OSAL Standard File APIs, 580
OS_remove
OSAL Standard File APIs, 581
OS_rename
OSAL Standard File APIs, 581
OS_rmdir
OSAL Directory APIs, 556
OS_rmfs
OSAL File System Level APIs, 592
OS_socket_prop_t, 941
creator, 941
name, 941
OS_stat
OSAL Standard File APIs, 582
OS_static_symbol_record_t, 942
Address, 942
Module, 942
Name, 943
OS_statvfs_t, 943
block_size, 943
blocks_free, 943
total_blocks, 944
OS_task_prop_t, 944
creator, 944
name, 945
priority, 945
stack_size, 945
OS_time_t, 945
ticks, 946
OS_timebase_prop_t, 946
accuracy, 947
creator, 947
freerun_time, 947
name, 947
nominal_interval_time, 947
OS_timer_prop_t, 948
accuracy, 948
creator, 948
interval_time, 948
name, 948
start_time, 949
OS_unmount
OSAL File System Level APIs, 593
OS_write
OSAL Standard File APIs, 585
OSAL BSP low level access APIs, 530
OS_BSP_GetArgC, 530
OS_BSP_GetArgV, 530
OS_BSP_GetResourceTypeConfig, 530
OS_BSP_SetExitCode, 530
OS_BSP_SetResourceTypeConfig, 530
OSAL Binary Semaphore APIs, 524
OS_BinSemCreate, 524
OS_BinSemDelete, 525
OS_BinSemFlush, 525
OS_BinSemGetIdByName, 526
OS_BinSemGetInfo, 527
OS_BinSemGive, 527
OS_BinSemTake, 528
OS_BinSemTimedWait, 528
OSAL Core Operation APIs, 543
OS_API_Init, 543
OS_API_Teardown, 544
OS_Application_Run, 544
OS_Application_Startup, 544
OS_ApplicationExit, 544
OS_ApplicationShutdown, 545
OS_DeleteAllObjects, 545
OS_IdleLoop, 545
OS_RegisterEventHandler, 546
OSAL Counting Semaphore APIs, 547
OS_CountSemCreate, 547
OS_CountSemDelete, 548
OS_CountSemGetIdByName, 549
OS_CountSemGetInfo, 549
OS_CountSemGive, 550
OS_CountSemTake, 550
OS_CountSemTimedWait, 551
OSAL Directory APIs, 553
OS_DirectoryClose, 553
OS_DirectoryOpen, 554
OS_DirectoryRead, 554
OS_DirectoryRewind, 555
OS_mkdir, 556
OS_rmdir, 556
OSAL Dynamic Loader and Symbol APIs, 607
OS_ModuleInfo, 607
OS_ModuleLoad, 608
OS_ModuleSymbolLookup, 608
OS_ModuleUnload, 609
OS_SymbolLookup, 610
OS_SymbolTableDump, 610
OSAL Error Info APIs, 569
OS_GetErrorName, 569
OS_StatusToInteger, 569
OSAL File Access Option Defines, 571
OS_READ_ONLY, 571
OS_READ_WRITE, 571
OS_WRITE_ONLY, 571
OSAL File System Level APIs, 586
OS_FS_GetPhysDriveName, 588
OS_FileSysAddFixedMap, 587
OS_FileSysStatVolume, 588
OS_GetFsInfo, 589
OS_TranslatePath, 593

OS_chkfs, 586
OS_initfs, 590
OS_mkfs, 591
OS_mount, 591
OS_rmfs, 592
OS_unmount, 593
OSAL Heap APIs, 595
OS_HeapGetInfo, 595
OSAL Message Queue APIs, 621
OS_QueueCreate, 621
OS_QueueDelete, 622
OS_QueueGet, 623
OS_QueueGetIdByName, 623
OS_QueueGetInfo, 624
OS_QueuePut, 624
OSAL Mutex APIs, 612
OS_MutSemCreate, 612
OS_MutSemDelete, 613
OS_MutSemGetIdByName, 613
OS_MutSemGetInfo, 614
OS_MutSemGive, 614
OS_MutSemTake, 615
OSAL Network ID APIs, 617
OS_NetworkGetHostName, 617
OS_NetworkGetID, 617
OSAL Object ID Utility APIs, 600
OS_ConvertToArrayIndex, 600
OS_ForEachObject, 601
OS_ForEachObjectOfType, 601
OS_GetResourceName, 602
OS_IdentifyObject, 603
OS_ObjectIdDefined, 603
OS_ObjectIdEqual, 604
OS_ObjectIdFromInteger, 604
OS_ObjectIdToArrayIndex, 605
OS_ObjectIdToInteger, 605
OSAL Object Type Defines, 596
OS_OBJECT_TYPE_OS_BINSEM, 596
OS_OBJECT_TYPE_OS_CONSOLE, 596
OS_OBJECT_TYPE_OS_COUNTSEM, 597
OS_OBJECT_TYPE_OS_DIR, 597
OS_OBJECT_TYPE_OS_FILESYS, 597
OS_OBJECT_TYPE_OS_MODULE, 597
OS_OBJECT_TYPE_OS_MUTEX, 597
OS_OBJECT_TYPE_OS_QUEUE, 598
OS_OBJECT_TYPE_OS_STREAM, 598
OS_OBJECT_TYPE_OS_TASK, 598
OS_OBJECT_TYPE_OS_TIMEBASE, 598
OS_OBJECT_TYPE_OS_TIMECB, 598
OS_OBJECT_TYPE_UNDEFINED, 599
OS_OBJECT_TYPE_USER, 599
OSAL Printf APIs, 619
OS_printf, 619
OS_printf_disable, 619
OS_printf_enable, 620
OSAL Real Time Clock APIs, 531
OS_GetLocalTime, 531
OS_SetLocalTime, 532
OS_TimeAdd, 533
OS_TimeAssembleFromMicroseconds, 533
OS_TimeAssembleFromMilliseconds, 534
OS_TimeAssembleFromNanoseconds, 534
OS_TimeAssembleFromSubseconds, 535
OS_TimeGetFractionalPart, 536
OS_TimeGetMicrosecondsPart, 536
OS_TimeGetMillisecondsPart, 537
OS_TimeGetNanosecondsPart, 538
OS_TimeGetSubsecondsPart, 539
OS_TimeGetTotalMicroseconds, 539
OS_TimeGetTotalMilliseconds, 540
OS_TimeGetTotalNanoseconds, 540
OS_TimeGetTotalSeconds, 541
OS_TimeSubtract, 542
OSAL Reference Point For Seek Offset Defines, 572
OS_SEEK_CUR, 572
OS_SEEK_END, 572
OS_SEEK_SET, 572
OSAL Return Code Defines, 558
OS_ERR_BAD_ADDRESS, 560
OS_ERR_FILE, 560
OS_ERR_INCORRECT_OBJ_STATE, 560
OS_ERR_INCORRECT_OBJ_TYPE, 561
OS_ERR_INVALID_ARGUMENT, 561
OS_ERR_INVALID_ID, 561
OS_ERR_INVALID_PRIORITY, 561
OS_ERR_INVALID_SIZE, 561
OS_ERR_NAME_NOT_FOUND, 562
OS_ERR_NAME_TAKEN, 562
OS_ERR_NAME_TOO_LONG, 562
OS_ERR_NO_FREE_IDS, 562
OS_ERR_NOT_IMPLEMENTED, 562
OS_ERR_OBJECT_IN_USE, 563
OS_ERR_OPERATION_NOT_SUPPORTED, 563
OS_ERR_OUTPUT_TOO_LARGE, 563
OS_ERR_SEM_NOT_FULL, 563
OS_ERR_STREAM_DISCONNECTED, 563
OS_ERROR_ADDRESS_MISALIGNED, 564
OS_ERROR_TIMEOUT, 564
OS_ERROR, 564
OS_FS_ERR_DEVICE_NOT_FREE, 564
OS_FS_ERR_DRIVE_NOT_CREATED, 564
OS_FS_ERR_NAME_TOO_LONG, 565
OS_FS_ERR_PATH_INVALID, 565
OS_FS_ERR_PATH_TOO_LONG, 565
OS_INVALID_INT_NUM, 565
OS_INVALID_POINTER, 565
OS_INVALID_SEM_VALUE, 566
OS_QUEUE_EMPTY, 566

OS_QUEUE_FULL, 566
OS_QUEUE_ID_ERROR, 566
OS_QUEUE_INVALID_SIZE, 566
OS_QUEUE_TIMEOUT, 567
OS_SEM_FAILURE, 567
OS_SEM_TIMEOUT, 567
OS_SUCCESS, 567
OS_TIMER_ERR_INTERNAL, 567
OS_TIMER_ERR_INVALID_ARGS, 568
OS_TIMER_ERR_TIMER_ID, 568
OS_TIMER_ERR_UNAVAILABLE, 568
OSAL Select APIs, 626
OS_SelectFdAdd, 626
OS_SelectFdClear, 627
OS_SelectFdIsSet, 627
OS_SelectFdZero, 628
OS_SelectMultiple, 628
OS_SelectSingle, 629
OSAL Semaphore State Defines, 523
OS_SEM_EMPTY, 523
OS_SEM_FULL, 523
OSAL Shell APIs, 631
OS_ShellOutputToFile, 631
OSAL Socket Address APIs, 632
OS_SocketAddrFromString, 632
OS_SocketAddrGetPort, 633
OS_SocketAddrInit, 633
OS_SocketAddrSetPort, 634
OS_SocketAddrToString, 635
OSAL Socket Management APIs, 636
OS_SocketAccept, 636
OS_SocketBind, 638
OS_SocketConnect, 639
OS_SocketGetIdByName, 639
OS_SocketGetInfo, 640
OS_SocketOpen, 641
OS_SocketRecvFrom, 641
OS_SocketSendTo, 642
OS_SocketShutdown, 643
OSAL Standard File APIs, 573
OS_CloseAllFiles, 575
OS_CloseFileByName, 575
OS_FDGetInfo, 576
OS_FileOpenCheck, 577
OS_OpenCreate, 579
OS_TimedRead, 583
OS_TimedWrite, 584
OS_chmod, 573
OS_close, 574
OS_cp, 576
OS_lseek, 577
OS_mv, 578
OS_read, 580
OS_remove, 581
OS_rename, 581
OS_stat, 582
OS_write, 585
OSAL Task APIs, 644
OS_TaskCreate, 644
OS_TaskDelay, 645
OS_TaskDelete, 646
OS_TaskExit, 646
OS_TaskFindIdBySystemData, 646
OS_TaskGetId, 647
OS_TaskGetIdByName, 647
OS_TaskGetInfo, 648
OS_TaskInstallDeleteHandler, 649
OS_TaskSetPriority, 649
OSAL Time Base APIs, 651
OS_TimeBaseCreate, 651
OS_TimeBaseDelete, 652
OS_TimeBaseGetFreeRun, 653
OS_TimeBaseGetIdByName, 654
OS_TimeBaseGetInfo, 654
OS_TimeBaseSet, 655
OSAL Timer APIs, 657
OS_TimerAdd, 657
OS_TimerCreate, 659
OS_TimerDelete, 660
OS_TimerGetIdByName, 661
OS_TimerGetInfo, 662
OS_TimerSet, 663
OSAL_API_VERSION
osapi-version.h, 1643
OSAL_BLOCKCOUNT_C
common_types.h, 1595
OSAL_INDEX_C
common_types.h, 1596
OSAL_OBJTYPE_C
common_types.h, 1596
OSAL_PRIORITY_C
osapi-task.h, 1636
OSAL_SIZE_C
common_types.h, 1596
OSAL_STACKPTR_C
osapi-task.h, 1636
OSAL_STATUS_C
common_types.h, 1596
OSAL_TASK_STACK_ALLOCATE
osapi-task.h, 1636
OSALMajorVersion
CFE_ES_HousekeepingTlm_Payload, 690
OSALMinorVersion
CFE_ES_HousekeepingTlm_Payload, 690
OSALMissionRevision
CFE_ES_HousekeepingTlm_Payload, 690
OSALRevision
CFE_ES_HousekeepingTlm_Payload, 691

object_ids
OS_FdSet, 928
ObjectName
CFE_TBL_FileDef, 810
ObjectSize
CFE_TBL_FileDef, 810
Offset
CFE_TBL_File_Hdr, 809
OnEvent
CFE_FS_FileWriteMetaData, 761
OneHzAdjust
CFE_TIME_DiagnosticTlm_Payload, 845
OneHzDirection
CFE_TIME_DiagnosticTlm_Payload, 845
OneTimeAdjust
CFE_TIME_DiagnosticTlm_Payload, 846
OneTimeDirection
CFE_TIME_DiagnosticTlm_Payload, 846
OpenFileCount
fm_cmd_utils.c, 1048
OpenFilesList
FM_OpenFilesPkt_t, 918
OpenFilesPkt
FM_GlobalData_t, 909
Opts
CFE_SB_PipeInfoEntry, 780
os_dirent_t, 927
FileName, 927
os_err_name_t
osapi-error.h, 1614
os_fsinfo_t, 930
FreeFds, 930
FreeVolumes, 930
MaxFds, 930
MaxVolumes, 931
os_fstat_t, 931
FileModeBits, 932
FileSize, 932
FileTime, 932
osal/docs/src/osal_frontpage.dox, 1593
osal/docs/src/osal_fs.dox, 1593
osal/docs/src/osal_timer.dox, 1593
osal/src/os/inc/common_types.h, 1593
osal/src/os/inc/osapi-binsem.h, 1601
osal/src/os/inc/osapi-bsp.h, 1602
osal/src/os/inc/osapi-clock.h, 1603
osal/src/os/inc/osapi-common.h, 1605
osal/src/os/inc/osapi-constants.h, 1608
osal/src/os/inc/osapi-countsem.h, 1609
osal/src/os/inc/osapi-dir.h, 1610
osal/src/os/inc/osapi-error.h, 1611
osal/src/os/inc/osapi-file.h, 1615
osal/src/os/inc/osapi-filesys.h, 1619
osal/src/os/inc/osapi-heap.h, 1621
osal/src/os/inc/osapi-idmap.h, 1621
osal/src/os/inc/osapi-macros.h, 1623
osal/src/os/inc/osapi-module.h, 1626
osal/src/os/inc/osapi-mutex.h, 1627
osal/src/os/inc/osapi-network.h, 1628
osal/src/os/inc/osapi-printf.h, 1629
osal/src/os/inc/osapi-queue.h, 1629
osal/src/os/inc/osapi-select.h, 1630
osal/src/os/inc/osapi-shell.h, 1631
osal/src/os/inc/osapi-sockets.h, 1631
osal/src/os/inc/osapi-task.h, 1634
osal/src/os/inc/osapi-timebase.h, 1638
osal/src/os/inc/osapi-timer.h, 1639
osal/src/os/inc/osapi-version.h, 1640
osal/src/os/inc/osapi.h, 1645
osal_blockcount_t
common_types.h, 1598
osal_id_t
common_types.h, 1598
osal_index_t
common_types.h, 1598
osal_objtype_t
common_types.h, 1599
osal_priority_t
osapi-task.h, 1637
osal_stackptr_t
osapi-task.h, 1637
osal_status_t
common_types.h, 1599
osal_task
osapi-task.h, 1637
osapi-common.h
OS_Event_t, 1607
OS_EventHandler_t, 1606
osapi-constants.h
OS_CHECK, 1608
OS_MAX_LOCAL_PATH_LEN, 1608
OS_OBJECT_CREATOR_ANY, 1608
OS_OBJECT_ID_UNDEFINED, 1609
OS_PEND, 1609
osapi-dir.h
OS_DIRENTRY_NAME, 1611
osapi-error.h
OS_ERROR_NAME_LENGTH, 1614
os_err_name_t, 1614
osapi-file.h
OS_FILESTAT_EXEC, 1616
OS_FILESTAT_ISDIR, 1617
OS_FILESTAT_MODE, 1617
OS_FILESTAT_READ, 1617
OS_FILESTAT_SIZE, 1617
OS_FILESTAT_TIME, 1618
OS_FILESTAT_WRITE, 1618
OS_file_flag_t, 1619

osapi-filesys.h
OS_CHK_ONLY, 1620
OS_REPAIR, 1620
osapi-idmap.h
OS_OBJECT_INDEX_MASK, 1623
OS_OBJECT_TYPE_SHIFT, 1623
osapi-macros.h
ARGCHECK, 1624
BUGCHECK, 1624
BUGREPORT, 1625
LENGTHCHECK, 1625
osapi-module.h
OS_MODULE_FLAG_GLOBAL_SYMBOLS, 1626
OS_MODULE_FLAG_LOCAL_SYMBOLS, 1627
osapi-select.h
OS_StreamState_t, 1630
osapi-sockets.h
OS_SOCKADDR_MAX_LEN, 1633
OS_SocketDomain_t, 1633
OS_SocketShutdownMode_t, 1634
OS_SocketType_t, 1634
osapi-task.h
OS_FP_ENABLED, 1636
OS_MAX_TASK_PRIORITY, 1636
OSAL_PRIORITY_C, 1636
OSAL_STACKPTR_C, 1636
OSAL_TASK_STACK_ALLOCATE, 1636
osal_priority_t, 1637
osal_stackptr_t, 1637
osal_task, 1637
osapi-timebase.h
OS_TimerSync_t, 1639
osapi-timer.h
OS_TimerCallback_t, 1640
osapi-version.h
OS_BUILD_BASELINE, 1641
OS_BUILD_NUMBER, 1641
OS_GetBuildNumber, 1644
OS_GetVersionCodeName, 1644
OS_GetVersionNumber, 1644
OS_GetVersionString, 1645
OS_MAJOR_VERSION, 1641
OS_MINOR_VERSION, 1641
OS_MISSION_REV, 1642
OS_REVISION, 1642
OS_STR_HELPER, 1642
OS_STR, 1642
OS_VERSION_CODENAME, 1643
OS_VERSION_STRING, 1643
OS_VERSION, 1642
OSAL_API_VERSION, 1643
osconfig-example.h
OS_BUFFER_MSG_DEPTH, 1145
OS_BUFFER_SIZE, 1145
OS_FS_DEV_NAME_LEN, 1146
OS_FS_PHYS_NAME_LEN, 1146
OS_FS_VOL_NAME_LEN, 1146
OS_MAX_API_NAME, 1146
OS_MAX_BIN_SEMAPHORES, 1146
OS_MAX_CMD_LEN, 1147
OS_MAX_CONSOLES, 1147
OS_MAX_COUNT_SEMAPHORES, 1147
OS_MAX_FILE_NAME, 1147
OS_MAX_FILE_SYSTEMS, 1148
OS_MAX_MODULES, 1148
OS_MAX_MUTEXES, 1148
OS_MAX_NUM_OPEN_DIRS, 1148
OS_MAX_NUM_OPEN_FILES, 1149
OS_MAX_PATH_LEN, 1149
OS_MAX_QUEUES, 1149
OS_MAX_SYM_LEN, 1150
OS_MAX_TASKS, 1150
OS_MAX_TIMEBASES, 1150
OS_MAX_TIMERS, 1151
OS_MODULE_FILE_EXTENSION, 1151
OS_PRINTF_CONSOLE_NAME, 1151
OS_QUEUE_MAX_DEPTH, 1151
OS_SHELL_CMD_INPUT_FILE_NAME, 1152
OS_SOCKADDR_MAX_LEN, 1152
OS_UTILITYTASK_PRIORITY, 1152
OS_UTILITYTASK_STACK_SIZE, 1152
OutputPort
CFE_EVS_HousekeepingTlm_Payload, 745
Overwrite
FM_CopyFileCmd_t, 877
FM_MoveFileCmd_t, 915
OwnerAppName
CFE_TBL_TblRegPacket_Payload, 833

PSPMajorVersion
CFE_ES_HousekeepingTlm_Payload, 693
PSPMinorVersion
CFE_ES_HousekeepingTlm_Payload, 694
PSPMissionRevision
CFE_ES_HousekeepingTlm_Payload, 694
PSPRevision
CFE_ES_HousekeepingTlm_Payload, 694
PacketFiles
FM_DirListPkt_t, 887
PacketID
CFE_EVS_LongEventTlm_Payload, 749
CFE_EVS_ShortEventTlm_Payload, 757
Padding
CFE_EVS_AppTlmData, 737
Parameter
CFE_TBL_NotifyCmd_Payload, 826
Path
OS_file_prop_t, 929

Payload
CFE_ES_AppNameCmd, 673
CFE_ES_DeleteCDSCmd, 679
CFE_ES_DumpCDSRegistryCmd, 681
CFE_ES_FileNameCmd, 682
CFE_ES_HousekeepingTlm, 684
CFE_ES_MemStatsTlm, 700
CFE_ES_OneAppTlm, 702
CFE_ES_OverWriteSysLogCmd, 703
CFE_ES_ReloadAppCmd, 707
CFE_ES_RestartCmd, 708
CFE_ES_SendMemPoolStatsCmd, 710
CFE_ES_SetMaxPRCountCmd, 712
CFE_ES_SetPerfFilterMaskCmd, 714
CFE_ES_SetPerfTriggerMaskCmd, 716
CFE_ES_StartApp, 718
CFE_ES_StartPerfDataCmd, 722
CFE_ES_StopPerfDataCmd, 724
CFE_EVS_AppNameBitMaskCmd, 728
CFE_EVS_AppNameCmd, 730
CFE_EVS_AppNameEventIDCmd, 732
CFE_EVS_AppNameEventIDMaskCmd, 734
CFE_EVS_BitMaskCmd, 739
CFE_EVS_HousekeepingTlm, 741
CFE_EVS_LongEventTlm, 748
CFE_EVS_SetEventFormatModeCmd, 754
CFE_EVS_SetLogModeCmd, 756
CFE_EVS_ShortEventTlm, 756
CFE_EVS_WriteAppDataFileCmd, 758
CFE_EVS_WriteLogDataFileCmd, 759
CFE_SB_AllSubscriptionsTlm, 765
CFE_SB_HousekeepingTlm, 767
CFE_SB_RouteCmd, 783
CFE_SB_SingleSubscriptionTlm, 787
CFE_SB_StatsTlm, 789
CFE_SB_WriteFileInfoCmd, 797
CFE_TBL_AbortLoadCmd, 799
CFE_TBL_ActivateCmd, 801
CFE_TBL_DeleteCDSCmd, 804
CFE_TBL_DumpCmd, 805
CFE_TBL_DumpRegistryCmd, 807
CFE_TBL_HousekeepingTlm, 812
CFE_TBL_LoadCmd, 822
CFE_TBL_NotifyCmd, 825
CFE_TBL_SendRegistryCmd, 827
CFE_TBL_TableRegistryTlm, 829
CFE_TBL_ValidateCmd, 835
CFE_TIME_DiagnosticTlm, 837
CFE_TIME_HousekeepingTlm, 851
CFE_TIME_OneHzAdjustmentCmd, 857
CFE_TIME_SetLeapSecondsCmd, 859
CFE_TIME_SetSignalCmd, 860
CFE_TIME_SetSourceCmd, 861
CFE_TIME_SetStateCmd, 862
CFE_TIME_TimeCmd, 867
CFE_TIME_ToneDataCmd, 869
PeakMemInUse
CFE_SB_StatsTlm_Payload, 793
PeakMsgIdsInUse
CFE_SB_StatsTlm_Payload, 793
PeakPipesInUse
CFE_SB_StatsTlm_Payload, 793
PeakQueueDepth
CFE_SB_PipeDepthStats, 777
CFE_SB_PipeInfoEntry, 780
PeakSBBuffersInUse
CFE_SB_StatsTlm_Payload, 793
PeakSubscriptionsInUse
CFE_SB_StatsTlm_Payload, 794
PerfDataCount
CFE_ES_HousekeepingTlm_Payload, 691
PerfDataEnd
CFE_ES_HousekeepingTlm_Payload, 691
PerfDataStart
CFE_ES_HousekeepingTlm_Payload, 691
PerfDataToWrite
CFE_ES_HousekeepingTlm_Payload, 692
PerfFilterMask
CFE_ES_HousekeepingTlm_Payload, 692
PerfMode
CFE_ES_HousekeepingTlm_Payload, 692
PerfState
CFE_ES_HousekeepingTlm_Payload, 692
PerfTriggerCount
CFE_ES_HousekeepingTlm_Payload, 693
PerfTriggerMask
CFE_ES_HousekeepingTlm_Payload, 693
Pipe
CFE_SB_RouteCmd_Payload, 784
CFE_SB_SingleSubscriptionTlm_Payload, 788
CFE_SB_SubEntries, 796
PipeDepthStats
CFE_SB_StatsTlm_Payload, 794
PipeId
CFE_SB_PipeDepthStats, 778
CFE_SB_PipeInfoEntry, 780
CFE_SB_RoutingFileEntry, 786
PipeName
CFE_SB_PipeInfoEntry, 780
CFE_SB_RoutingFileEntry, 786
PipeOptsErrorCounter
CFE_SB_HousekeepingTlm_Payload, 772
PipeOverflowErrorCounter
CFE_SB_HousekeepingTlm_Payload, 772
PipesInUse
CFE_SB_StatsTlm_Payload, 794
PktSegment
CFE_SB_AllSubscriptionsTlm_Payload, 766

PoolHandle
CFE_ES_PoolStatsTlm_Payload, 706
CFE_ES_SendMemPoolStatsCmd_Payload, 711
PoolSize
CFE_ES_MemPoolStats, 699
PoolStats
CFE_ES_PoolStatsTlm_Payload, 706
Priority
CFE_ES_AppInfo, 671
CFE_ES_StartAppCmd_Payload, 720
CFE_ES_TaskInfo, 725
CFE_SB_Qos_t, 782
priority
OS_task_prop_t, 945
ProcessorID
CFE_EVS_PacketID, 751
CFE_FS_Header, 763
ProcessorResets
CFE_ES_HousekeepingTlm_Payload, 693
psp/fsw/inc/cfe_psp.h, 1646
Ptr
CFE_ES_PoolAlign, 705

Qos
CFE_SB_SingleSubscriptionTlm_Payload, 788
CFE_SB_SubEntries, 796

RegisteredCoreApps
CFE_ES_HousekeepingTlm_Payload, 694
RegisteredExternalApps
CFE_ES_HousekeepingTlm_Payload, 695
RegisteredLibs
CFE_ES_HousekeepingTlm_Payload, 695
RegisteredTasks
CFE_ES_HousekeepingTlm_Payload, 695
Reliability
CFE_SB_Qos_t, 782
Reserved
CFE_TBL_File_Hdr, 809
ResetSubtype
CFE_ES_HousekeepingTlm_Payload, 695
ResetType
CFE_ES_HousekeepingTlm_Payload, 696
ResourceId
CFE_ES_AppInfo, 671
RestartType
CFE_ES_RestartCmd_Payload, 709

SBBuffersInUse
CFE_SB_StatsTlm_Payload, 794
SIZE_BYTE
cfe_psp.h, 1657
SIZE_HALF
cfe_psp.h, 1657
SIZE_WORD
cfe_psp.h, 1658
sample_mission_cfg.h
CFE_MISSION_ES_APP_TLM_MSG, 1220
CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN, 1220
CFE_MISSION_ES_CDS_MAX_NAME_LENGTH, 1221
CFE_MISSION_ES_CMD_MSG, 1221
CFE_MISSION_ES_CRC_16, 1222
CFE_MISSION_ES_CRC_32, 1222
CFE_MISSION_ES_CRC_8, 1222
CFE_MISSION_ES_DEFAULT_CRC, 1222
CFE_MISSION_ES_HK_TLM_MSG, 1223
CFE_MISSION_ES_MAX_APPLICATIONS, 1223
CFE_MISSION_ES_MEMSTATS_TLM_MSG, 1223
CFE_MISSION_ES_PERF_MAX_IDS, 1224
CFE_MISSION_ES_POOL_MAX_BUCKETS, 1224
CFE_MISSION_ES_SEND_HK_MSG, 1224
CFE_MISSION_EVS_CMD_MSG, 1225
CFE_MISSION_EVS_HK_TLM_MSG, 1225
CFE_MISSION_EVS_LONG_EVENT_MSG_MSG, 1225
CFE_MISSION_EVS_MAX_MESSAGE_LENGTH, 1225
CFE_MISSION_EVS_SEND_HK_MSG, 1226
CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG, 1226
CFE_MISSION_MAX_API_LEN, 1226
CFE_MISSION_MAX_FILE_LEN, 1226
CFE_MISSION_MAX_PATH_LEN, 1227
CFE_MISSION_SB_ALLSUBS_TLM_MSG, 1227
CFE_MISSION_SB_CMD_MSG, 1228
CFE_MISSION_SB_HK_TLM_MSG, 1228
CFE_MISSION_SB_MAX_PIPES, 1228
CFE_MISSION_SB_MAX_SB_MSG_SIZE, 1228
CFE_MISSION_SB_ONESUB_TLM_MSG, 1229
CFE_MISSION_SB_SEND_HK_MSG, 1229
CFE_MISSION_SB_STATS_TLM_MSG, 1229
CFE_MISSION_SB_SUB_RPT_CTRL_MSG, 1229
CFE_MISSION_TBL_CMD_MSG, 1230
CFE_MISSION_TBL_HK_TLM_MSG, 1230
CFE_MISSION_TBL_MAX_FULL_NAME_LEN, 1230
CFE_MISSION_TBL_MAX_NAME_LENGTH, 1230
CFE_MISSION_TBL_REG_TLM_MSG, 1231
CFE_MISSION_TBL_SEND_HK_MSG, 1231
CFE_MISSION_TEST_CMD_MSG, 1231
CFE_MISSION_TEST_HK_TLM_MSG, 1231
CFE_MISSION_TIME_1HZ_CMD_MSG, 1232
CFE_MISSION_TIME_AT_TONE_WAS, 1232
CFE_MISSION_TIME_AT_TONE_WILL_BE, 1232
CFE_MISSION_TIME_CFG_DEFAULT_TAI, 1232
CFE_MISSION_TIME_CFG_DEFAULT_UTC, 1233
CFE_MISSION_TIME_CFG_FAKE_TONE, 1233

CFE_MISSION_TIME_CMD_MSG, 1233
CFE_MISSION_TIME_DATA_CMD_MSG, 1234
CFE_MISSION_TIME_DEF_DELAY_SECS, 1234
CFE_MISSION_TIME_DEF_DELAY_SUBS, 1234
CFE_MISSION_TIME_DEF_LEAPS, 1234
CFE_MISSION_TIME_DEF_MET_SECS, 1234
CFE_MISSION_TIME_DEF_MET_SUBS, 1235
CFE_MISSION_TIME_DEF_STCF_SECS, 1235
CFE_MISSION_TIME_DEF_STCF_SUBS, 1235
CFE_MISSION_TIME_DIAG_TLM_MSG, 1235
CFE_MISSION_TIME_EPOCH_DAY, 1236
CFE_MISSION_TIME_EPOCH_HOUR, 1236
CFE_MISSION_TIME_EPOCH_MINUTE, 1236
CFE_MISSION_TIME_EPOCH_SECOND, 1236
CFE_MISSION_TIME_EPOCH_YEAR, 1236
CFE_MISSION_TIME_FS_FACTOR, 1236
CFE_MISSION_TIME_HK_TLM_MSG, 1237
CFE_MISSION_TIME_MAX_ELAPSED, 1237
CFE_MISSION_TIME_MIN_ELAPSED, 1237
CFE_MISSION_TIME_SEND_CMD_MSG, 1238
CFE_MISSION_TIME_SEND_HK_MSG, 1238
CFE_MISSION_TIME_TONE_CMD_MSG, 1238
sample_perfids.h
CFE_MISSION_ES_MAIN_PERF_ID, 1239
CFE_MISSION_ES_PERF_EXIT_BIT, 1240
CFE_MISSION_EVS_MAIN_PERF_ID, 1240
CFE_MISSION_SB_MAIN_PERF_ID, 1240
CFE_MISSION_SB_MSG_LIM_PERF_ID, 1240
CFE_MISSION_SB_PIPE_OFLOW_PERF_ID, 1240
CFE_MISSION_TBL_MAIN_PERF_ID, 1241
CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID, 1241
CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID, 1241
CFE_MISSION_TIME_MAIN_PERF_ID, 1241
CFE_MISSION_TIME_SENDMET_PERF_ID, 1241
CFE_MISSION_TIME_TONE1HZISR_PERF_ID, 1242
CFE_MISSION_TIME_TONE1HZTASK_PERF_ID, 1242
SearchOpenFileData
fm_cmd_utils.c, 1047
Seconds
CFE_TIME_OneHzAdjustmentCmd_Payload, 858
CFE_TIME_SysTime, 866
CFE_TIME_TimeCmd_Payload, 868
Seconds1HzAdj
CFE_TIME_HousekeepingTlm_Payload, 853
SecondsDelay
CFE_TIME_HousekeepingTlm_Payload, 853
SecondsMET
CFE_TIME_HousekeepingTlm_Payload, 854
SecondsSTCF
CFE_TIME_HousekeepingTlm_Payload, 854
SendErrors
CFE_SB_PipeInfoEntry, 781
Sequence
CCSDS_PrimaryHeader, 665
ServerFlyState
CFE_TIME_DiagnosticTlm_Payload, 846
Size
CFE_ES_CDSRegDumpRec, 678
CFE_TBL_Info, 821
CFE_TBL_TblRegPacket_Payload, 833
Source
FM_CopyFileCmd_t, 877
FM_DecompressCmd_t, 879
FM_MoveFileCmd_t, 915
FM_RenameFileCmd_t, 920
Source1
FM_ChildQueueEntry_t, 873
FM_ConcatCmd_t, 875
Source2
FM_ChildQueueEntry_t, 874
FM_ConcatCmd_t, 875
SpacecraftID
CFE_EVS_PacketID, 752
CFE_FS_Header, 763
Spare
CFE_ES_TaskInfo, 726
CFE_EVS_AppNameBitMaskCmd_Payload, 729
CFE_EVS_BitMaskCmd_Payload, 740
CFE_EVS_SetEventFormatCode_Payload, 753
CFE_EVS_SetLogMode_Payload, 755
CFE_SB_PipeDepthStats, 778
CFE_SB_PipeInfoEntry, 781
CFE_SB_RouteCmd_Payload, 784
FM_FileInfoPkt_t, 890
FM_HousekeepingPkt_t, 914
Spare01
FM_GetDirFileCmd_t, 895
FM_GetDirPktCmd_t, 897
Spare1
CFE_EVS_HousekeepingTlm_Payload, 745
CFE_EVS_LongEventTlm_Payload, 749
Spare2
CFE_EVS_HousekeepingTlm_Payload, 745
CFE_EVS_LongEventTlm_Payload, 749
Spare2Align
CFE_SB_HousekeepingTlm_Payload, 772
Spare3
CFE_EVS_HousekeepingTlm_Payload, 746
Spare8a
FM_GlobalData_t, 909
Spare8b
FM_GlobalData_t, 909
stack_size
OS_task_prop_t, 945

StackSize
CFE_ES_AppInfo, 671
CFE_ES_StartAppCmd_Payload, 720
CFE_ES_TaskInfo, 726
start_time
OS_timer_prop_t, 949
StartAddress
CFE_ES_AppInfo, 672
State
CFE_SB_RoutingFileEntry, 786
FM_TableEntry_t, 924
StreamId
CCSDS_PrimaryHeader, 665
SubType
CFE_FS_Header, 763
CFE_SB_SingleSubscriptionTlm_Payload, 789
SubscribeErrorCounter
CFE_SB_HousekeepingTlm_Payload, 772
SubscriptionsInUse
CFE_SB_StatsTlm_Payload, 795
Subseconds
CFE_TIME_OneHzAdjustmentCmd_Payload, 858
CFE_TIME_SysTime, 866
Subsecs1HzAdj
CFE_TIME_HousekeepingTlm_Payload, 854
SubsecsDelay
CFE_TIME_HousekeepingTlm_Payload, 854
SubsecsMET
CFE_TIME_HousekeepingTlm_Payload, 855
SubsecsSTCF
CFE_TIME_HousekeepingTlm_Payload, 855
Subsystem
CCSDS_ExtendedHeader, 664
SuccessValCounter
CFE_TBL_HousekeepingTlm_Payload, 818
SysLogBytesUsed
CFE_ES_HousekeepingTlm_Payload, 696
SysLogEntries
CFE_ES_HousekeepingTlm_Payload, 696
SysLogMode
CFE_ES_HousekeepingTlm_Payload, 696
SysLogSize
CFE_ES_HousekeepingTlm_Payload, 697
SystemId
CCSDS_ExtendedHeader, 664

Table
CFE_ES_CDSRegDumpRec, 678
TableEntryIndex
FM_SetTableStateCmd_t, 923
TableEntryState
FM_SetTableStateCmd_t, 923
TableLoadedOnce
CFE_TBL_Info, 821
CFE_TBL_TblRegPacket_Payload, 834
TableName
CFE_TBL_AbortLoadCmd_Payload, 800
CFE_TBL_ActivateCmd_Payload, 802
CFE_TBL_DelCDSCmd_Payload, 803
CFE_TBL_DumpCmd_Payload, 806
CFE_TBL_File_Hdr, 809
CFE_TBL_FileDef, 811
CFE_TBL_SendRegistryCmd_Payload, 828
CFE_TBL_ValidateCmd_Payload, 836
Target
FM_ChildQueueEntry_t, 874
FM_ConcatCmd_t, 876
FM_CopyFileCmd_t, 877
FM_DecompressCmd_t, 879
FM_MoveFileCmd_t, 915
FM_RenameFileCmd_t, 920
TaskId
CFE_ES_TaskInfo, 726
TaskName
CFE_ES_TaskInfo, 726
TelemetryHeader
CFE_ES_HousekeepingTlm, 684
CFE_ES_MemStatsTlm, 700
CFE_ES_OneAppTlm, 702
CFE_EVS_HousekeepingTlm, 741
CFE_EVS_LongEventTlm, 748
CFE_EVS_ShortEventTlm, 757
CFE_SB_AllSubscriptionsTlm, 765
CFE_SB_HousekeepingTlm, 767
CFE_SB_SingleSubscriptionTlm, 787
CFE_SB_StatsTlm, 790
CFE_TBL_HousekeepingTlm, 812
CFE_TBL_TableRegistryTlm, 829
CFE_TIME_DiagnosticTlm, 837
CFE_TIME_HousekeepingTlm, 851
TgtFilename
CFE_TBL_FileDef, 811
ticks
OS_time_t, 946
TimeOfLastUpdate
CFE_TBL_Info, 821
CFE_TBL_TblRegPacket_Payload, 834
TimeSeconds
CFE_FS_Header, 764
TimeSinceTone
CFE_TIME_DiagnosticTlm_Payload, 846
TimeSource
CFE_TIME_SourceCmd_Payload, 864
TimeSubSeconds
CFE_FS_Header, 764
TlmHeader
FM_DirListPkt_t, 887
FM_FileInfoPkt_t, 891

FM_FreeSpacePkt_t, 892
FM_HousekeepingPkt_t, 914
FM_OpenFilesPkt_t, 918
ToneDataCounter
CFE_TIME_DiagnosticTlm_Payload, 847
ToneDataLatch
CFE_TIME_DiagnosticTlm_Payload, 847
ToneIntCounter
CFE_TIME_DiagnosticTlm_Payload, 847
ToneIntErrorCounter
CFE_TIME_DiagnosticTlm_Payload, 847
ToneMatchCounter
CFE_TIME_DiagnosticTlm_Payload, 848
ToneMatchErrorCounter
CFE_TIME_DiagnosticTlm_Payload, 848
ToneOverLimit
CFE_TIME_DiagnosticTlm_Payload, 848
ToneSignalCounter
CFE_TIME_DiagnosticTlm_Payload, 848
ToneSignalLatch
CFE_TIME_DiagnosticTlm_Payload, 849
ToneSource
CFE_TIME_SignalCmd_Payload, 863
ToneTaskCounter
CFE_TIME_DiagnosticTlm_Payload, 849
ToneUnderLimit
CFE_TIME_DiagnosticTlm_Payload, 849
total_blocks
OS_statvfs_t, 944
TotalFiles
FM_DirListPkt_t, 888
TotalSegments
CFE_SB_AllSubscriptionsTlm_Payload, 766
TriggerMask
CFE_ES_SetPerfTrigMaskCmd_Payload, 717
TriggerMaskNum
CFE_ES_SetPerfTrigMaskCmd_Payload, 717
TriggerMode
CFE_ES_StartPerfCmd_Payload, 721
Type
CFE_ES_AppInfo, 672

uint16
common_types.h, 1599
uint32
common_types.h, 1599
uint64
common_types.h, 1599
uint8
common_types.h, 1600
UnmarkedMem
CFE_SB_HousekeepingTlm_Payload, 773
UnregisteredAppCounter
CFE_EVS_HousekeepingTlm_Payload, 746
User
OS_file_prop_t, 929
UserDefAddr
CFE_TBL_Info, 821

valid
OS_module_address_t, 935
ValidationCounter
CFE_TBL_HousekeepingTlm_Payload, 818
ValidationFuncPtr
CFE_TBL_TblRegPacket_Payload, 834
Value
CFE_SB_MsgId_t, 775
value
OS_bin_sem_prop_t, 926
OS_count_sem_prop_t, 927
VersionCounter
CFE_TIME_DiagnosticTlm_Payload, 849
VirtualMET
CFE_TIME_DiagnosticTlm_Payload, 850